RESEARCH REPORT

IRIT/RR–2017–01–FR

A Formal Study of Boolean Games with Random Formulas as Pay Functions

Erik Martin-Dorel and Sergei Soloviev

IRIT (U. Toulouse, CNRS, INPT, UPS, UT1, UT2J),
Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 9, France

erik.martin-dorel@irit.fr
sergei.soloviev@irit.fr

February 5, 2017

Abstract

In this paper, we present a probabilistic analysis of Boolean games. We consider the class of Boolean games where pay functions are given by random Boolean formulas. This permits to study certain properties of this class in its totality, such as the probability of existence of a winning strategy, including its asymptotic behaviour. With the help of the Coq proof assistant, we develop a Coq library of Boolean games, to provide a formal proof of our results, and a basis for further developments.

Keywords

Boolean games. Random process. Coq formal proofs.

Contents

1 Introduction

One of the main motivations to consider the classes of games with random parameters is that it is a good method to explore these classes in their totality and to understand the relative importance of various properties of games (such as simultaneous or alternating moves, different assumptions concerning the access to information, etc.)

The situation when, for a given game, only its type can be known in advance but its parameters cannot, is common when the game-theoretic approach is used to study the behaviour of embedded systems, i.e., when at least some of the players are programs and the parameters are not fully controlled. The augmented frequency of interaction that usually surpasses any conceivable capacity of human players, and rapid evolution of parameters of interaction makes the use of probabilistic methods quite natural.

In this report we present the first results obtained via this approach applied to Boolean Games [11, 10, 7, 3]. More precisely, we limit our study to Boolean Games with random formulas that represent pay functions. This model is naturally related to the situation when games between automated systems (e.g., embedded systems in computer networks) are considered. Indeed, assuming that the players are finite non-deterministic machines, they can be simulated by a family of Boolean formulas.

Before the exploration may start, several choices have to be done concerning the probabilistic model.

Regarding elementary events: we have chosen Boolean functions as elementary events. Another possible choice would be to consider formulas as syntactic objects, but the former choice makes it easier to define probability distributions that are naturally related to the properties on the associated Boolean games (while the latter choice would require to cope with the complex behaviour of the logical equivalence of formulas).

It seems natural also to consider a probability space for each n, where n is the number of variables. Indeed, n is one of the key parameters involved when considering the complexity of Boolean functions, and if we shall need to consider different values of n, it is possible to combine in some way the spaces built for each n.

Let us recall some some basic properties of Boolean functions.

(i) The domain of these functions is $\mathbf{2}^n = \{0,1\}^n$, the set of all Boolean vectors of length n (which contains 2^n elements).

(ii) Each Boolean function can be identified with a characteristic function of a subset of $\mathbf{2}^n$ and thus with the subset itself, so the set of all elementary events is $\Omega = \mathbf{2}^{\mathbf{2}^n}$.

(iii) A subset of $\mathbf{2}^n$ may be identified with a formula of n variables in the full disjunctive normal form[1] that is satisfied by exactly these vectors (to each vector corresponds the conjunction of variables and their negations: to 1 at the i-th place corresponds v_i and to 0 corresponds $\neg v_i$).

(iv) The set Ω is a complete Boolean algebra, and logical operators on elements of Ω correspond to the set-theoretic operators on $\mathbf{2}^n$. The top element of this algebra is the set $\mathbf{2}^n \in \Omega$ (it represents the Boolean function *true*) and the bottom element of this algebra is the set $\emptyset \in \Omega$ (it represents the Boolean function *false*). How these logical operators "interact" with probability on Ω is a separate question, for example, $\mathbb{P}(true)$ needs not of course be equal to 1.

(v) There is a natural partial order on the elements of Ω, that is defined by the inclusion of subsets of $\mathbf{2}^n$ and at the same time by Boolean implication (see Definition 1 below).

Since the elements of Ω may be seen at the same time as Boolean functions and as sets of Boolean vectors, we pose the following definition:

[1]By convention, the empty DNF (or constant 0) corresponds to the empty subset.

Definition 1. Let $\omega_1, \omega_2 \in \Omega = \mathbf{2}^{\mathbf{2}^n}$. We shall write $\omega_1 \Rightarrow_0 \omega_2$ and say that ω_2 is true on ω_1 if ω_2 is true (as a Boolean function) on all elements of ω_1. It is equivalent to the inclusion of subsets of $\mathbf{2}^n$:

$$\forall \omega_1, \omega_2 \in \Omega.\ (\omega_1 \Rightarrow_0 \omega_2) \iff (\omega_1 \subseteq \omega_2)\,.$$

Remark. We may see random Boolean functions also as (not necessarily independent) vectors of 2^n random variables with values in $\mathbf{2} = \{0,1\}$.

Regarding the sigma-algebra of events: as usual for finite probability spaces, we consider the sigma-algebra $\mathcal{S}$ of all subsets of Ω:

$$\mathcal{S} = \mathbf{2}^{\mathbf{2}^{\mathbf{2}^n}}\,.$$

Regarding the probability distributions on the spaces of Boolean formulas: as it is noticed in [8], it is often assumed that all Boolean function on a given number of variables have the same probability (see also [15]). In this report where we start our study, we decided to consider a slightly more general class of probability distributions, where Boolean functions are generated by a Bernoulli scheme on Boolean vectors, with any probability p as parameter. Some more sophisticated ways to define probability distributions on Boolean expressions are discussed in [8] and we plan to explore them in a near future.

In Section 2, we prove several general results on the probability of winning strategies assuming an arbitrary probability distribution. Then in Section 3, we study the case of Boolean formulas constructed through a finite Bernoulli process, and specialise our results in this simpler setting. In Section 4, we further study the probability that a winning strategy exists for player A, with the new assumption that player A knows s bits of the opponent player. Then in Section 5, we study the asymptotic behaviour of the aforementioned probability, with respect to the knowledge of the second player's choices. Section 6 will be devoted to technical remarks about the formalisation of our results within the Coq [5] formal proof assistant.

All the results of the report up to Section 4 have been formally verified within Coq.[2] An archive of the Coq code, along with some documentation, is available at the URL `https://sourcesup.renater.fr/coq-bool-games/`

Besides that it is interesting in itself to the type theory community, the formal certification is nowadays common in the development and characterization of the behaviour of autonomous programs, which is one of the subjects of this study.

2 Probability of Winning Strategies

Building upon the material of the previous section, we can consider any probability $\mathbb{P}$ defined on the sigma-algebra $\mathcal{S} = \mathbf{2}^{\mathbf{2}^{\mathbf{2}^n}}$, and thus obtain a probability space $(\Omega, \mathcal{S}, \mathbb{P})$. We shall show in this section that several results can be derived in this setting, however general as it may sound.

Example (using Definition 1)**.** The probability of the event "ω is true on ω_0", with fixed $\omega_0 \in \Omega$, is

$$\mathbb{P}(\omega_0 \Rightarrow_0 \omega) = \sum_{\substack{\omega \\ \omega_0 \Rightarrow_0 \omega}} \mathbb{P}(\{\omega\}).$$

Below we shall consider the Boolean Games of two players A and B with a random Boolean function F of n variables as the pay function of A, and its negation as the pay function for B. We shall assume that A controls the first k variables, and B the remaining $n - k$ variables.

[2]In the sequel of the report, all definitions and theorems will be stated in mathematical syntax, and the corresponding Coq identifier will be given between brackets.

The strategy of A is any vector that belongs to $\mathbf{2}^k$ (valuation of the first k variables) and the strategy of B any valuation of the remaining $n-k$ variables (a vector of $\mathbf{2}^{n-k}$).

The outcome of the game is given by player A's pay function $F : \mathbf{2}^n \to \mathbf{2}$, which can thereby be viewed as a function $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \to \mathbf{2}$ (mapping a profile strategy to a Boolean outcome). In the sequel of the report, we shall identify these two possible types for the function F — while in the formal development they will be encoded respectiviely as `(bool_fun n)` and `(bool_game n k)`.

Definition 2 (`winA`)**.** For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \to \mathbf{2}$, a strategy $a = (a_1, \ldots, a_k)$ of player A is winning if it wins against any strategy $b \in \mathbf{2}^{n-k}$ of B:

$$\mathrm{win}_A[F](a) := \forall b \in \mathbf{2}^{n-k}.\ F(a,b) = 1.$$

If there is no ambiguity, we shall omit the name of the game and simply write $\mathrm{win}_A(a)$.

In other words, a is winning if the pay function is equal to 1 on all vectors of length n that "extend" a. This led us to introduce the following

Definition 3 (`w_`, `W_`)**.** For any $a \in \mathbf{2}^k$, let ω_a be the set of vectors in $\mathbf{2}^n$ that extend a:

$$\omega_a := \{v \in \mathbf{2}^n \mid v_1 = a_1 \wedge \cdots \wedge v_k = a_k\} \in \Omega$$

and W_a be the set of all Boolean functions that are true on ω_a:

$$W_a := \{\omega \in \Omega \mid \omega_a \Rightarrow_0 \omega\} \in \mathcal{S}.$$

These definitions straightforwardly imply the following lemma:

Lemma 1 (`winA_eq`)**.** For any Boolean function $F : \mathbf{2}^n \to \mathbf{2}$ and any strategy $a \in \mathbf{2}^k$ of player A in the associated Boolean game, we have:

$$\mathrm{win}_A(a) \iff F \in W_a.$$

Lemma 1 implies that the probability that a winning strategy exists satisfies:

$$\mathbb{P}(\exists a.\, \mathrm{win}_A(a)) = \mathbb{P}\left(\bigcup_{a \in \mathbf{2}^k} W_a\right). \tag{1}$$

Then we shall rely on the inclusion-exclusion formula, which we proved in full generality as follows:

Theorem 2 (`Pr_bigcup_incl_excl`)**.** For any finite probability space $(\Omega, \mathcal{S}, \mathbb{P})$ and any sequence of events $(S_i)_{0 \le i < n}$, we have:

$$\mathbb{P}\left(\bigcup_{0 \le i < n} S_i\right) = \sum_{m=1}^{n} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbb{N} \cap [0,n) \\ \mathrm{Card}\, J = m}} \mathbb{P}\left(\bigcap_{j \in J} S_j\right). \tag{2}$$

Proof. For proving this theorem in Coq we formalise a small theory of indicator functions $\mathrm{Ind}_S : \Omega \to \{0,1\}$ for any finite set $S \subseteq \Omega$, including the fact that the expectation satisfies $\mathbb{E}(\mathrm{Ind}_S) = \mathbb{P}(S)$, then formalise an algebraic proof of the inclusion-exclusion formula.[3]
These proofs strongly rely on the `bigop` theory of the `MathComp` library, as well as on the tactic "`under`" that we developed in Ltac to easily "rewrite under lambdas" (e.g., under the $\sum$ symbol). These tactics facilities will be further detailed in Section 6. □

[3]taking inspiration from the proof path presented at https://en.wikipedia.org/wiki/Inclusion-exclusion_principle#Algebraic_proof

Hence the following result:

Theorem 3 (`Pr_ex_winA`)**.** For any finite probability space $(\Omega, \mathcal{S}, \mathbb{P})$, the probability that there exists some strategy $a = (a_1, \dots, a_k)$ of A that is winning satisfies:

$$\begin{aligned}\mathbb{P}(\exists a.\,\mathrm{win}_A(a)) &= \sum_{a \in \mathbf{2}^k} \mathbb{P}(W_a) - \sum_{\substack{a,a' \in \mathbf{2}^k \\ a \neq a'}} \mathbb{P}(W_a \cap W_{a'}) + \cdots \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \operatorname{Card} J = m}} \mathbb{P}\left(\bigcap_{a \in J} W_a\right).\end{aligned}$$

Proof. The result follows from (1) and (2), after reindexing all big-operators $\bigcup$, $\sum$, $\bigcap$ by natural numbers instead of Boolean vectors $a \in \mathbf{2}^k$, or conversely. □

Theorem 3 is applicable with *any* probability $\mathbb{P}$, but it is not easy to handle. Below we shall investigate in more detail the case when $\mathbb{P}$ is relatively simple.

3 Bernoulli Process and Winning Strategies

In this section we still consider the space $\Omega = \mathbf{2}^{\mathbf{2}^n}$ of random Boolean formulas of n variables endowed with the discrete σ-algebra $\mathcal{S} = \mathbf{2}^{\mathbf{2}^{\mathbf{2}^n}}$, and the associated Boolean games with parameter $0 \leq k \leq n$. But now, we assume that the Boolean formulas (in DNF) are determined by a random choice of the Boolean vectors that satisfy the formulas.

To be more precise, we assume the probability that each vector $v \in \mathbf{2}^n$ belongs to the truth-set of the formula F is equal to p, $(0 \leq p \leq 1)$. As usual, we notate $q = 1 - p$.

In the sequel, we shall often identify Boolean functions $F : \mathbf{2}^n \to \mathbf{2}$ and their truth-set $|F| \in \mathbf{2}^{\mathbf{2}^n}$. In the Coq formalisation, the distinction between the two is always made explicit, and the function that gives the truth-set of a Boolean function is implemented by a function

```
finset_of_bool_fun : ∀n : nat, bool_fun n -> {set bool_vec n}
```

Our setup amounts to constructing a Bernoulli process, that is a series of independent Bernoulli trials, to decide whether each vector $v \in \mathbf{2}^n$ belongs to $|F|$ or not. The construction of the DNF is immediate (it is formalised as a function `DNF_of`, which is the inverse of `finset_of_bool_fun`).

We obtain the following result:

Lemma 4 (`dist_BernoulliE`)**.** For any $F \in \Omega$, the probability of an elementary event $\{F\}$ with respect to the considered probability $\mathbb{P}_{n;p}$ (modelling a series of 2^n independent Bernoulli trials of parameter p) is:

$$\mathbb{P}_{n;p}(\{F\}) = p^k (1-p)^{2^n - k}$$

where k denotes the number of vectors in the truth-set of F, and $2^n - k$ denotes the number of vectors in the truth-set of the negation of F.

Proof. The proof (and its formal counterpart in Coq) straightforwardly derives from the definitions. □

For now, we assume that the choices of player A and B are done simultaneously. A wins if the value of F is 1, otherwise B wins. What is the probability that A has a winning strategy?

First, suppose that the strategy a of A is fixed, and let us compute the probability that it is winning. We first prove the following

Lemma 5 (`Pr_implies0_Bern`)**.** Let $S \subseteq \mathbf{2}^n$, and let us notate $m := \operatorname{Card} S$. Then the probability that F is true on S satisfies: $\mathbb{P}(S \Rightarrow_0 F) = p^m$.

Proof. We follow the following proof path:

$$\begin{aligned}
\mathbb{P}_{n;p}(S \Rightarrow_0 F) &= \sum_{\substack{F \\ S \subseteq F}} \mathbb{P}_{n;p}(\{F\}) \\
&= \sum_{\substack{S' \subseteq \mathbf{2}^n \setminus S \\ F = S \cup S'}} \mathbb{P}_{n;p}(\{F\}) \\
&= \sum_{S' \subseteq \mathbf{2}^n \setminus S} p^{\operatorname{Card}(S \cup S')} q^{2^n - \operatorname{Card}(S \cup S')} \text{ by Lemma 4} \\
&= \sum_{m'=0}^{2^n - m} \binom{2^n - m}{m'} p^{m+m'} q^{2^n - m - m'} \\
&= p^m \sum_{m'=0}^{2^n - m} \binom{2^n - m}{m'} p^{m'} q^{(2^n - m) - m'} \\
&= p^m (p+q)^{2^n - m} \\
&= p^m.
\end{aligned}$$

□

Lemma 6 (`card_w_a_Bern`)**.** For any strategy a of player A, we have

$$\operatorname{Card} w_a = 2^{n-k}.$$

Proof. This lemma easily follows from the fact that w_a is the image of the strategy space $\mathbf{2}^{n-k}$ of B by an injective function. □

Hence the following theorem, which gives the probability that a fixed strategy of A is winning:

Theorem 7 (`Pr_winA_Bern`)**.** For any strategy a of player A, we have

$$\mathbb{P}_{n;p}(\operatorname{win}_A(a)) = p^{2^{n-k}}.$$

Proof. This result is an immediate consequence of Lemmas 5 and 6. □

Now, let us determine what is the probability that A has at least one winning strategy. One may first notice the following

Lemma 8 (`w_trivIset`)**.** The truth-sets $|w_a|$ (for $a \in J \subset \mathbf{2}^k$) are pairwise disjoint.

Proof. By contradiction: if we had $a, b \in \mathbf{2}^k$ such that $w_a \neq w_b$ and $w_a \cap w_b \neq \emptyset$, then let us pose $x \in w_a \cap w_b$. By unfolding Definition 3, this means that the first k bits of x coincides with all bits of a, and likewise for b. This implies that $a = b$ and thereby $w_a = w_b$, which contradicts the initial hypothesis. □

Lemma 8 implies that we have

$$\operatorname{Card}\left(\bigcup_{a \in J} w_a\right) = \sum_{a \in J} \operatorname{Card} w_a = \operatorname{Card} J \cdot 2^{n-k}. \tag{3}$$

We can now prove the following

Theorem 9 (`Pr_ex_winA_Bern`)**.** For any n and k, if $\mathbb{P}$ follows the Bernoulli scheme that we previously constructed, the probability that player A has a winning strategy is:

$$\mathbb{P}_{n;p}(\exists a.\, \mathrm{win}_A(a)) = 1 - \left(1 - p^{2^{n-k}}\right)^{2^k}. \tag{4}$$

Proof. Thanks to Theorem 3, we can write:

$$\begin{aligned}
\mathbb{P}_{n;p}(\exists a.\, \mathrm{win}_A(a)) &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \mathrm{Card}\, J = m}} \mathbb{P}_{n;p}\left(\bigcap_{a \in J} W_a\right) \\
&= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \mathrm{Card}\, J = m}} \mathbb{P}_{n;p}\left(\bigcap_{a \in J} [w_a \Rightarrow_0 F]\right) \\
&= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \mathrm{Card}\, J = m}} \mathbb{P}_{n;p}\left[\left(\bigcup_{a \in J} w_a\right) \Rightarrow_0 F\right] \\
&= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \mathrm{Card}\, J = m}} p^{(\mathrm{Card}(\bigcup_{a \in J} w_a))} \text{ by Lemma 5} \\
&= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \mathrm{Card}\, J = m}} p^{m \cdot 2^{n-k}} \text{ by using (3) and Lemma 6} \\
&= \sum_{m=1}^{2^k} (-1)^{m-1} \binom{2^k}{m} p^{m \cdot 2^{n-k}} \\
&= 1 - \sum_{m=0}^{2^k} \binom{2^k}{m} (-p^{2^{n-k}})^m 1^{2^k - m} \\
&= 1 - \left(1 - p^{2^{n-k}}\right)^{2^k}.
\end{aligned}$$

□

By duality, one can derive the existence of a winning strategy for player B:

Corollary 10 (`Pr_ex_winB_Bern`)**.** For any p, n, k, if $\mathbb{P}_{n;p}$ denotes the considered Bernoulli scheme (with parameters $0 \le p \le 1$ and $n \in \mathbb{N}$) and if k denotes the number of variables controlled by player A, then the probability that player B has a winning strategy is:

$$\mathbb{P}_{n;p}(\exists b.\, \mathrm{win}_B(b)) = 1 - \left(1 - (1 - p)^{2^k}\right)^{2^{n-k}}.$$

Proof. We first define $\mathrm{win}_B(b)$ as the predicate "$\forall a \in \mathbf{2}^k.\ F(a,b) = 0$," then show that $\mathrm{win}_B(b)$ holds if and only if in the dual game associated with function $F' := (b,a) \mapsto \neg F(a,b)$, the first player wins with strategy b (denoting by $n - k$ the number of variables that he controls). The rest of the Coq proof consists in relating the properties of the dual game with respect to the original game and finally applying Theorem 9. □

3.1 Discussion

The computations above may seem elementary, but lead to some observations that are less trivial. As we may see, there is a considerable probability that there is no winning strategy at all. For example, if $p = 1/2, n = 4, k = 2$ the probability that there is no winning strategy for A is $1-(1-1/16)^4 \approx 0.228$ and (by symmetry) the same for B.

One may notice also that when $p \neq 0$ is fixed, $k = c \cdot n$ for a given constant $0 < c < 1$, and n tending towards $+\infty$, the probability of existence of winning strategy for A and B tends to 0.

If (for some F) there is no winning strategy nor for A, neither for B, then the order of moves becomes important. Indeed, let a be an arbitrary strategy of A. Since it is not winning, there exists at least one b of B such that $(a,b) \notin |F|$. If B makes his choice after A, he always may win. Similarly, if A makes his choice after B, he may win.

Bradfield, Gutierrez and Wooldridge notice [4] (as do some other authors): "As they are conventionally formulated, Boolean games assume that players make their choices in ignorance of the choices being made by other players—they are games of simultaneous moves. For many settings, this is clearly unrealistic." Our simple probabilistic analysis provides a direct quantitative argument to support this general observation.

4 Partial Information on the Opponent's Choices

Now, let us consider the case when A may have partial information about the choices of B before making his own choice. Without loss of generality, we may assume that he knows the values of the first s variables among the variables $v_{k+1},\dots,v_n$ controlled by B. We shall consider the probability of existence of strategies of A such that for every vector $b_{1:s} = (b_1,\dots,b_s) \in \mathbf{2}^s$ there exists a strategy $a \in \mathbf{2}^k$ that wins against any strategy $b \in \mathbf{2}^{n-k}$ where first s values coincide with $b_{1:s}$.

In other words, we are interested in the probability of guaranteed win by A when s choices by B among $n-k$ are known (assuming $0 \leq s \leq n-k$). We thus introduce the following predicate:

Definition 4 (`winA_knowing`)**.** For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \to \mathbf{2}$ and any $b_{1:s} \in \mathbf{2}^s$, we say that a strategy $a \in \mathbf{2}^k$ is winning under the knowledge of $b_{1:s}$ if it is winning against all strategy profile $(a,b) \in \mathbf{2}^k \times \mathbf{2}^{n-k}$ that is compatible with $b_{1:s}$:

$$\mathrm{win}_A(a \mid b_{1:s}) := \forall b \in \mathbf{2}^{n-k}.\ \texttt{compat\_knowing}(b_{1:s},b) \implies F(a,b) = 1,$$

where

$$\texttt{compat\_knowing}(b_{1:s},b) := \forall i \in \mathbf{2}^s.\ (b_{1:s})_i = b_i.$$

For relating this predicate with that of Definition 2, the proof of the following lemma is immediate:

Lemma 11 (`winA_knowingE`)**.** For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \to \mathbf{2}$ and any bit-vectors $b_{1:s} \in \mathbf{2}^s$ and $a \in \mathbf{2}^k$, we have:

$$\mathrm{win}_A[F](a \mid b_{1:s}) = \mathrm{win}_A[\mathrm{bgk}(F,b_{1:s})](a)$$

where $\mathrm{bgk}(F,b_{1:s}) : \mathbf{2}^k \times \mathbf{2}^{n-s-k}$ is the Boolean game defined by:

$$\mathrm{bgk}(F,b_{1:s})(a,b') = F(a,(b_{1:s},b')).$$

Now, to compute the probability $\mathbb{P}_{n;p}\left(\forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A(a \mid b_{1:s})\right)$ in the space $(\Omega,\mathcal{S},\mathbb{P}_{n;p})$ introduced in Section 3, we shall first construct a product space $(\Omega',\mathcal{S}',\mathbb{P}')$ and show that it is isomorphic to $(\Omega,\mathcal{S},\mathbb{P}_{n;p})$.

First, we note that there are 2^s possible Boolean vectors $b_{1:s} = (b_1, \dots, b_s)$ and for all $b_{1:s}$, we pose

$$B_{b_{1:s}} = \{v \in \mathbf{2}^n \mid v_{k+1} = b_1 \wedge \cdots \wedge v_{k+s} = b_s\}.$$

The family $(B_{b_{1:s}})_{b_{1:s} \in \mathbf{2}^s}$ constitutes a partition of $\mathbf{2}^n$ (we have $\mathbf{2}^n = \bigcup_{b_{1:s} \in \mathbf{2}^s} B_{b_{1:s}}$, intersections of $B_{b_{1:s}}$ for different $b_{1:s}$ are empty, and no set $B_{b_{1:s}}$ is empty).

Second, we define $\Omega_{b_{1:s}} := \mathbf{2}^{B_{b_{1:s}}}$ as the powerset of $B_{b_{1:s}}$ and show that there is a one-to-one correspondance between $\Omega_{b_{1:s}}$ and $\mathbf{2}^{\mathbf{2}^{n-s}}$. We shall denote the corresponding bijections by $g : \Omega_{b_{1:s}} \to \mathbf{2}^{\mathbf{2}^{n-s}}$ and $h : \mathbf{2}^{\mathbf{2}^{n-s}} \to \Omega_{b_{1:s}}$. In the formal development, the related lemmas are named `bool_fun_of_OmegaB_bij` and `OmegaB_of_bool_fun_bij`.

Next, we consider the probability $\mathbb{P}_{b_{1:s}} := \mathbb{P}_{n-s;\,p} \circ h^{-1}$ defined as the pushforward distribution (with respect to function h) of the Bernoulli process $\mathbb{P}_{n-s;\,p}$ with parameters $n-s$ and p.

We then consider the product space $(\Omega', \mathcal{S}', \mathbb{P}')$ defined by:

$$\begin{cases} \Omega' = \prod_{b_{1:s} \in \mathbf{2}^s} \Omega_{b_{1:s}} \\ \mathcal{S}' = \mathbf{2}^{\Omega'} \\ \mathbb{P}' = \bigotimes_{b_{1:s} \in \mathbf{2}^s} \mathbb{P}_{b_{1:s}} \end{cases}$$

Relying on functions g and h, we finally show that there is a one-to-one correspondance between Ω' and $\Omega = \mathbf{2}^{\mathbf{2}^n}$. We shall denote the corresponding bijections by $g' : \Omega' \to \Omega$ and $h' : \Omega \to \Omega'$. In the formal development, the related lemmas are named `bool_fun_of_Omega'_bij` and `Omega'_of_bool_fun_bij`.

We now prove that the spaces $(\Omega, \mathcal{S}, \mathbb{P}_{n;\,p})$ and $(\Omega', \mathcal{S}', \mathbb{P}')$ are isomorphic:

Lemma 12 (`isom_dist_Omega'`)**.** The probability distribution $\mathbb{P}_{n;\,p}$ (defined in Section 3 as the Bernoulli process with parameters n and p) is extensionally equal to the pushforward distribution of $\mathbb{P}'$ with respect to function g'.

Proof. In the Coq formal proof, this lemma amounts to splitting a big-operator expression with respect to the partition of $\mathbf{2}^n$, reindexing big-operator expressions half-a-dozen times, and rewriting "cancellation lemmas" for simplifying the composition of a bijection and its inverse function. Also, the use of our `under` tactic (see Section 6) contributed to simplify the mechanisation of this proof. □

A key ingredient for the sequel will be the following

Lemma 13 (`ProductDist.indep`)**.** Given a finite type I and a family of finite probability spaces $(\Omega_i, \mathcal{S}_i = \mathbf{2}^{\Omega_i}, \mathbb{P}_i)_{i \in I}$, the product space defined by

$$\begin{cases} \Omega_\Pi = \prod_{i \in I} \Omega_i \\ \mathcal{S}_\Pi = \mathbf{2}^{\Omega_\Pi} \\ \mathbb{P}_\Pi = \bigotimes_{i \in I} \mathbb{P}_i \end{cases}$$

is such that the projections $(\pi_i : \Omega_\Pi \to \Omega_i)_{i \in I}$ are independent random variables. In other words, for any family of events $(Q_i)_{i \in I} \in \prod_{i \in I} \mathcal{S}_i$, we have:

$$\mathbb{P}_\Pi\left(\bigcap_{i \in I} \pi_i^{-1}(Q_i)\right) = \prod_{i \in I} \mathbb{P}_i(Q_i).$$

We can now prove the following

Theorem 14 (`Pr_ex_winA_knowing_Bern`)**.** For all $p \in [0,1]$ and for all integers n, k, s satisfying $0 \le s \le n-k \le n$, if $\mathbb{P}_{n;p}$ is the Bernoulli process with parameters n and p defined in Section 3, the probability of guaranteed win for player A knowing s choices of player B among his $n-k$ variables is:

$$\mathbb{P}_{n;p}\left(\forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A(a \mid b_{1:s})\right) = \left(1 - \left(1 - p^{2^{n-k-s}}\right)^{2^k}\right)^{2^s}.$$

Proof. We follow the following proof path:

$$\begin{aligned}&\mathbb{P}_{n;p}\left(\forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A(a \mid b_{1:s})\right)\\ &= \mathbb{P}_{n;p}\left\{F \in \Omega \,\middle|\, \forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[F](a \mid b_{1:s})\right\}\end{aligned}$$

hence by using Lemma 11

$$= \mathbb{P}_{n;p}\left\{F \in \Omega \,\middle|\, \forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[\mathrm{bgk}(F, b_{1:s})](a)\right\}$$

hence by using Lemma 12

$$= (\mathbb{P}' \circ g'^{-1})\left\{F \in \Omega \,\middle|\, \forall b_{1:s} \in \mathbf{2}^s.\ \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[\mathrm{bgk}(F, b_{1:s})](a)\right\}$$

hence by using elementary facts on g, g' and the bgk function defined in Lemma 11

$$= \mathbb{P}'\left\{f \in \Omega' \,\middle|\, \forall b_{1:s} \in \mathbf{2}^s.\ f(b_{1:s}) \in \left\{S \in \Omega_{b_{1:s}} \,\middle|\, \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[g(S)](a)\right\}\right\}$$

hence by using Lemma 13

$$= \prod_{b_{1:s} \in \mathbf{2}^s} \mathbb{P}_{b_{1:s}}\left\{S \in \Omega_{b_{1:s}} \,\middle|\, \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[g(S)](a)\right\}$$

hence by definition of $\mathbb{P}_{b_{1:s}}$

$$= \prod_{b_{1:s} \in \mathbf{2}^s} \left(\mathbb{P}_{n-s;p} \circ h^{-1}\right)\left\{S \in \Omega_{b_{1:s}} \,\middle|\, \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[g(S)](a)\right\}$$

hence by definition of g and h

$$= \prod_{b_{1:s} \in \mathbf{2}^s} \mathbb{P}_{n-s;p}\left\{F \in \mathbf{2}^{2^{n-s}} \,\middle|\, \exists a \in \mathbf{2}^k.\ \mathrm{win}_A[F](a)\right\}$$

hence by using Theorem 9 in the case of random formulas with $n-s$ variables

$$\begin{aligned}&= \prod_{b_{1:s} \in \mathbf{2}^s} \left(1 - \left(1 - p^{2^{n-s-k}}\right)^{2^k}\right)\\ &= \left(1 - \left(1 - p^{2^{n-k-s}}\right)^{2^k}\right)^{2^s}.\end{aligned}$$

□

For example, if only 1 bit (i.e., 1 choice) of B is known, we get the probability

$$\left(1-\left(1-p^{2^{n-k-1}}\right)^{2^k}\right)^2 = 1-2\left(1-p^{2^{n-k-1}}\right)^{2^k}+\left(1-p^{2^{n-k-1}}\right)^{2^{k+1}}. \tag{5}$$

We may compare this probability with the probability of existence of unconditionally winning strategy in Section 3.

Let us notate $t = p^{2^{n-k-1}}$. Then $p^{2^{n-k}} = t^2$. The difference between the probabilities (4) and (5) is

$$\begin{aligned}
&1-2(1-t)^{2^k}+(1-t)^{2^{k+1}}-(1-(1-t^2)^{2^k})\\
&=(1-t^2)^{2^k}-2(1-t)^{2^k}+(1-t)^{2^{k+1}}\\
&=(1-t)^{2^k}(1+t)^{2^k}-2(1-t)^{2^k}+\left((1-t)^{2^k}\right)^2\\
&=(1-t)^{2^k}((1+t)^{2^k}-2+(1-t)^{2^k})\\
&=(1-t)^{2^k}\left(2\cdot\sum_{j=1}^{2^{k-1}}\binom{2^k}{2\cdot j}t^{2\cdot j}\right).
\end{aligned}$$

One may consider this value as the increase of probability of guaranteed win by one bit of information about strategy of B.

Remark. We considered above the *probability of guaranteed win*, i.e., the probability that for every choice of the values of s variables by B there exists a strategy for A that wins against all strategies of B with this fixed choice. This problem is purely combinatorial and does not depend on the "preferences" of B. In general this probability will be different from the probability of non-guaranteed win for a given strategy of A, as this probability will be influenced by the preferences of B for some choices, the dependency of these choices on F, and so on.

5 Probability of Guaranteed Win: Growth Rate

Using the result given by Theorem 14, we would like to study how the probability of guaranteed win grows with each bit of information concerning the choice of B.

First, we notice that when s tends to $n-k$ the probability of guaranteed win for A tends just to

$$\left(1-\left(1-p^{2^0}\right)^{2^k}\right)^{2^{n-k}} = \left(1-(1-p)^{2^k}\right)^{2^{n-k}} = 1-\underbrace{\left[1-\left(1-(1-p)^{2^k}\right)^{2^{n-k}}\right]}_{\text{proba. of guaranteed win for } B}$$

More interesting question may be: what is the order of growth of the difference

$$\left(1-\left(1-p^{2^{n-k-s}}\right)^{2^k}\right)^{2^s}-\left(1-\left(1-p^{2^{n-k}}\right)^{2^k}\right) \tag{6}$$

when s is small with respect to k and n? Below is a proof sketch for this question.

In the same way as in the observation at the end of the previous section, we notate $t = p^{2^{n-k-s}}$. By the binomial formula, we have:

$$1-(1-t)^{2^k} = 2^k t-(2^k t)^2\sum_{i=2}^{2^k}(-1)^i 2^{-2k}\binom{2^k}{i}t^{i-2}. \tag{7}$$

Let us find some rough estimate. We notice that if

$$2^k t < 1 \qquad \left(\text{that is, if} \quad 2^k p^{2^{n-k-s}} < 1\right) \tag{8}$$

then the absolute value of the $(i+1)$th member of the sum $\sum$ in (7) is less than that of the i-th member because it is obtained by multiplication by $((2^k - i)/(i+1))t < 2^k t$. So, if (8) holds, then the sum (positive) is less than its first term. Moreover, the first term of the sum $\sum$ in (7) is $2^{-2k}\frac{2^k(2^k-1)}{2} < \frac{1}{2}$. So, if (8) is satisfied for some n, k, s, then from (7) we obtain

$$1-(1-t)^{2^k} \geq 2^k t - \frac{1}{2}(2^k t)^2 = 2^k t\left(1-\frac{1}{2}2^k t\right) > 2^{k-1}t. \tag{9}$$

Thus, under these conditions

$$\left(1-\left(1-p^{2^{n-s-k}}\right)^{2^k}\right)^{2^s} > 2^{(k-1)2^s} p^{2^{n-k}}. \tag{10}$$

A note on condition (8): let us consider the stronger inequality

$$2^k t \leq \frac{1}{2}. \tag{11}$$

It is equivalent to

$$2^{n-k-s}\log_2 p \leq -(k+1).$$

Since $0 < p < 1$ we may write instead

$$2^{n-k-s}|\log_2 p| \geq (k+1).$$

Applying the log a second time, we obtain the equivalent condition

$$(n-k-s) \geq \log_2(k+1) - \log_2(|\log_2 p|). \tag{12}$$

For example, if $p = 1/2$, then it becomes $(n-k-s) > \log_2(k+1)$. And if $0 < p < 1/2$, the term $\log_2(|\log_2 p|)$ is positive and we have $\log_2(k+1) > \log_2(k+1) - \log_2(|\log_2 p|)$.

Next, a simple analysis (along these lines) applied to $(1-(1-p^{2^{n-k}})^{2^k})$ gives an estimation

$$\left(1-\left(1-p^{2^{n-k}}\right)^{2^k}\right) = 2^k p^{2^{n-k}} - \sum_{i=1}^{2^k}\binom{2^k}{i}\left(-p^{2^{n-k}}\right)^i < 2^k p^{2^{n-k}} \tag{13}$$

Combining (10) and (13) yields the following inequality:

$$\begin{aligned}\left(1-\left(1-p^{2^{n-s-k}}\right)^{2^k}\right)^{2^s} - \left(1-\left(1-p^{2^{n-k}}\right)^{2^k}\right) &> 2^{(k-1)2^s}p^{2^{n-k}} - 2^k p^{2^{n-k}} \\ &= \left(2^{(k-1)2^s} - 2^k\right)p^{2^{n-k}}.\end{aligned} \tag{14}$$

In other words, the order of growth is essentially given by $2^{(k-1)2^s}$.

6 Remarks on the Formal Setup in the Coq Proof Assistant

6.1 Related Works on Formal Libraries of Probability

There have been several works focusing on the formalisation of measure theory or probability using interactive theorem proving. Some of these works only deal with discrete probability, or focus on the analysis of randomised algorithms; others formalise large fragments of measure theory up to Lebesgue's integration theory.

Using the HOL proof assistant, Hurd [13] developed a framework for proving properties of randomised programs, relying on a formalisation of measure theory, and following a "monadic transformation" approach that provides the user with an infinite sequence of independent, identically distributed $Bernoulli(\frac{1}{2})$ random variables.

Still using the HOL proof assistant and building upon Hurd's work, Mhamdi, Hasan and Tahar [12, 14] developed a comprehensive formalisation of measure theory, including Lebesgue's integration theory.

Using the Coq proof assistant, Audebaud and Paulin-Mohring [2] developed the ALEA library[4] that provides a framework to reason about randomised functional programs. Unlike Hurd's approach, it does not require a complete formalisation of measure theory: it is built upon a Coq axiomatisation of the interval $[0,1]$ and it interprets randomised programs as (discrete) probability distributions.

Still using the Coq formal proof assistant, Affeldt, Hagiwara and Sénizergues [1] developed the Infotheo library[5] that provides a formalisation of information theory. This library comes with a formalisation of finite probability theory and strongly relies on the theories of the MathComp library[6].

For developing our library on random Boolean games, we have chosen to rely on the Infotheo library. Even though it only deals with finite probability, this setting was sufficient for formalising our results and, further, it allowed us to benefit from the facilities of the SSReflect/MathComp library. In the rest of this section, we shall summarise the main notions that we used from the MathComp library and present our related contributions (in Section 6.2), then describe the overall setup of the Infotheo probability theory and present our related contributions (in Section 6.3).

6.2 MathComp and Our Related Contributions

The MathComp library was born in the Mathematical Components project, which aimed at formalising the Odd Order Theorem in the Coq proof assistant [9], while organising formal proofs into components to get a reusable library of mathematical facts. It is built upon SSReflect, an extension of Coq's proof language that has a native support for the so-called small scale reflection (and in particular Boolean reflection) and often leads to concise proof scripts.

For our library of random Boolean games, we have been especially using the following libraries: (*i*) `fintype` for finite types with decidable equality, (*ii*) `finfun` for functions over finite domains, (*iii*) `finset` for finite sets, (*iv*) `bigop` for properties on "big-operators".

Big-operators and rewriting under lambdas Regarding big-operators such as $\sum$, $\prod$, $\bigcap$ or $\bigcup$, they are formalised in MathComp as a higher-order function `bigop` that takes several arguments, including a function that specifies the "domain predicate" and the "general term". For example,

[4] https://www.lri.fr/~paulin/ALEA/
[5] https://staff.aist.go.jp/reynald.affeldt/shannon
[6] https://math-comp.github.io/math-comp/

the sum

$$\sum_{\substack{i=0 \\ i \text{ odd}}}^{4} i^2$$

can be formally written as `\sum_(0 <= i < 5 | odd i) i^2`, which amounts to the following term if we get rid of the `bigop` notation:

```
bigop 0 (index_iota 0 5) (fun i:nat => BigBody i addn (odd i) (i^2))
```

If we want to transform such a big-operator expression by rewriting its domain predicate or general term, the following two MathComp lemmas on big-operators can be used.

```
eq_bigr :
  forall (R : Type) (idx : R) (op : R -> R -> R) (I : Type)
  (r : seq I) (P : pred I) (F1 F2 : I -> R),
  (forall i : I, P i -> F1 i = F2 i) ->
  \big[op/idx]_(i <- r | P i) F1 i = \big[op/idx]_(i <- r | P i) F2 i


eq_bigl :
  forall (R : Type) (idx : R) (op : R -> R -> R) (I : Type)
  (r : seq I) (P1 P2 : pred I) (F : I -> R),
  P1 =1 P2 ->
  \big[op/idx]_(i <- r | P1 i) F i = \big[op/idx]_(i <- r | P2 i) F i
```

Still, applying them directly would require to provide the entire term corresponding to the function we want to obtain.

We thus developed a tactic `under` for “rewriting under the lambdas” of big-operators.[7]

Here is an example of use: for a goal that looks like

```
  A : finType
  n : nat
  F : A -> nat
========================================================
  0 <= \sum_(0 <= k < n)
       \sum_(J in {set A} | #|J :&: [set: A]| == k)
       \sum_(j in J) F j
```

the proof script

```
under eq_bigr [k Hk] under eq_bigl [J] rewrite setIT.
```

will yield the following goal:

```
  A : finType
  n : nat
  F : A -> nat
========================================================
  0 <= \sum_(0 <= k < n)
       \sum_(J in {set A} | #|J| == k)
       \sum_(j in J) F j
```

[7]A generalized version of our tactic, also applicable for notions of MathComp such as matrices, polynomials, and so on, is available at `https://github.com/erikmd/ssr-under-tac` and we plan to submit it for possible inclusion in MathComp once the issue `https://github.com/math-comp/math-comp/issues/62` is resolved.

Dependent product of finTypes MathComp has built-in support for finite functions: for any `(A:finType)` and `(T:Type)`, the notation `{ffun A -> T}` stands for the type of finite functions from `A` to `T`. If n denotes the cardinal of `A`, these functions are represented by a n-tuple of elements of `T`, which allows one to obtain convenient properties such as the extensionality of finite functions, which wouldn't hold otherwise in the constructive, intensional logic of Coq.

If `T` is also a finite type, then the MathComp library allows one to automatically retrieve (thanks to type inference and so-called canonical structures) a finite type structure for the type `{ffun A -> T}` itself. Thus, this construct amounts to the non-dependent product of a `finType`.

However, for formalising our results and in particularly to construct the type Ω' that appears in Section 4, we have been led to formalise the dependent product of a finite family of finite types. This material is gathered in a file `fprod.v` which provides a type `fprod`, some notations in MathComp style and several support results such as lemmas `fprodP` and `fprodE`, whose signature is as follows:

```
fprod : forall I : finType, (I -> finType) -> finType

fprodP : forall (I : finType) (T_ : I -> finType) (f1 f2: fprod I T_),
         (forall x : I, f1 x = f2 x) <-> f1 = f2

fprodE : forall (I : finType) (T_ : I -> finType)
                (g : forall i : I, T_ i) (x : I),
         [fprod i => g i] x = g x
```

This theory involves proofs with dependent types, and to facilitate the formalisation process we tried to follow a MathComp formalisation style as much as possible, by using finite functions, records with Boolean conditions, and so on. This enabled us to rely on extensionality of functions, the Altenkirch-Streicher K axiom and proof irrelevance, which can be used "axiom-free" in the decidable fragment of MathComp `finTypes`.

6.3 Infotheo and Our Related Contributions

The Infotheo library relies on MathComp as well as the `Reals` theory from Coq's standard library. Among the Infotheo theories, the `proba` theory was the starting point of our formalisation. It first defines distributions as a dependent record `dist`, gathering a function `pmf` that gives the probability of each elementary event, and a proof that the sum of these probabilities is equal to 1:

```
Record dist (A : finType) :=
  mkDist { pmf :> A -> R+ ;
           pmf1 : \rsum_(a in A) pmf a = 1 }.
```

Then, it defines the probability of a subset of `A` as the sum of the probabilities of all elementary events in `A`:

```
Definition Pr (A : finType) (P : dist A) (E : {set A}) :=
  \rsum_(a in E) P a.
```

Then, basic properties of probability and expectation are provided in this setting.

On top of the Infotheo theories, we have developed the following contributions:
(*i*) a formalisation of the pushforward distribution `dist_img` with the associated lemma

```
Lemma Pr_dist_img :
  forall {A B : finType} (X : A -> B) (PA : dist A) (E : {set B}),
  Pr (dist_img X PA) E = Pr PA (X @^-1: E).
```

(*ii*) a formal proof of a general version of the inclusion–exclusion theorem that we presented above in Theorem 2; (*iii*) the product distribution of a family of distributions, whose signature is as follows:

```
ProductDist.d :
  forall (I : finType) (T_ : I -> finType),
  (forall i : I, dist (T_ i)) -> dist (fprod I T_)
```

The associated independence result was presented above as Lemma 13.

7 Conclusion

In this work, we used the basics of the theory of Boolean games. In this sense, our work is obviously related to this area. But to the best of our knowledge, the idea of using probability theory applied to a certain class of Boolean games as a whole (in difference from merely random strategies) is new. The analysis of the whole class of games permits to discover some quantitative properties of these games that would be difficult to discover in the study of an individual game.

Furthermore, we used type theory and interactive theorem proving to formalise our results in order to give strong guarantees on their correctness as well as to extend existing formal libraries with new items.

In particular, we have proved a closed formula for the probability of existence of winning strategies in those random Boolean games. We specialised this result with a probability distribution on Boolean functions that are generated by a Bernoulli scheme on Boolean vectors with any probability p as parameter (it can be noted that this setting includes the case where all Boolean functions have the same probability).

In this report our methods remained elementary, but they permitted to estimate the relative importance of the cases where the players use simultaneous and alternative moves. Another interesting phenomenon seems to us to be the growth of probability of the win as function of the information about the choices of the opponent. Essentially, it is much faster than usual 2^s (2^{2^s}) where s is the quantity of information (number of extra bits) known to the player. This phenomenon emphasises the difference between the information that is required for winning and the "measure of knowledge" of the opponent and its strategies.

We already mentioned the interest of machine checked verification for the games between autonomous programs (embedded systems).

As a future work, we plan to consider more general classes of probability distributions and explore the "weight" of information with respect to winning in this more general setting.

We plan also to consider more closely the connection with algorithmic games [6].

Acknowledgements

This work was partially supported by the FAGames project of LabEx CIMI.
The authors would like to thank Evgeny Dantsin and Jan-Georg Smaus for fruitful discussions on algorithmic games and other topics. The first author also wishes to thank Cyril Cohen for advice and enhancement suggestions for our `under` tactic.

References

[1] Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues. Formalization of Shannon's theorems. *J. Autom. Reasoning*, 53(1):63–103, 2014. doi:10.1007/s10817-013-9298-1.

[2] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Sci. Comput. Program.*, 74(8):568–589, 2009. doi:10.1016/j.scico.2007.09.002.

[3] Élise Bonzon. *Modélisation des interactions entre agents rationnels : les jeux booléens*. PhD thesis, Université Toulouse III – Paul Sabatier, Toulouse, France, November 2007.

[4] Julian C. Bradfield, Julian Gutierrez, and Michael Wooldridge. Partial-order Boolean games: informational independence in a logic-based model of strategic interaction. *Synthese*, 193(3):781–811, 2016. doi:10.1007/s11229-015-0991-y.

[5] The Coq Development Team. *The Coq Proof Assistant: Reference Manual: version 8.5*, 2016. URL: https://coq.inria.fr/distrib/V8.5pl2/files/Reference-Manual.pdf.

[6] Evgeny Dantsin, Jan-Georg Smaus, and Sergei Soloviev. Algorithms in Games Evolving in Time: Winning Strategies Based on Testing. In *Isabelle Users Workshop – ITP 2012*, 2012. 18 pages.

[7] Paul E. Dunne and Wiebe van der Hoek. Representation and complexity in boolean games. In José Júlio Alferes and João Alexandre Leite, editors, *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings*, volume 3229 of *Lecture Notes in Computer Science*, pages 347–359. Springer, 2004. doi:10.1007/978-3-540-30227-8_30.

[8] Danièle Gardy. Random Boolean expressions. In René David, Danièle Gardy, Pierre Lescanne, and Marek Zaionc, editors, *Computational Logic and Applications, CLA '05*, volume AF of *DMTCS Proceedings*, pages 1–36, Chambéry, France, 2006. Discrete Mathematics and Theoretical Computer Science.

[9] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013. URL: http://dx.doi.org/10.1007/978-3-642-39634-2_14, doi:10.1007/978-3-642-39634-2_14.

[10] Paul Harrenstein. *Logic in Conflict. Logical Explorations in Strategic Equilibrium*. PhD thesis, Utrecht University, September 2004.

[11] Paul Harrenstein, Wiebe van der Hoek, John-Jules Meyer, and Cees Witteveen. Boolean Games. In J. van Benthem, editor, *Proceedings of the 8th International Conference on Theoretical Aspects of Rationality and Knowledge (TARK'01)*, pages 287–298, San Francisco, 2001. Morgan Kaufmann.

[12] Osman Hasan and Sofiène Tahar. Using theorem proving to verify expectation and variance for discrete random variables. *J. Autom. Reasoning*, 41(3-4):295–323, 2008. doi:10.1007/s10817-008-9113-6.

[13] Joe Hurd. *Formal verification of probabilistic algorithms*. PhD thesis, University of Cambridge, 2002.

[14] Tarek Mhamdi, Osman Hasan, and Sofiène Tahar. On the formalization of the Lebesgue integration theory in HOL. In Matt Kaufmann and Lawrence C. Paulson, editors, *Interactive Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14,*

2010. Proceedings, volume 6172 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2010. doi:10.1007/978-3-642-14052-5_27.

[15] John Riordan and C. E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21:83–93, August 1942.

Institut de Recherche en Informatique de Toulouse

CNRS
INPT
UPS
UT1

ACADIE

Abstract

In this paper, we present a probabilistic analysis of Boolean games. We consider the class of Boolean games where pay functions are given by random Boolean formulas. This permits to study certain properties of this class in its totality, such as the probability of existence of a winning strategy, including its asymptotic behaviour. With the help of the Coq proof assistant, we develop a Coq library of Boolean games, to provide a formal proof of our results, and a basis for further developments.

Keywords

Boolean games. Random process. Coq formal proofs.