



HAL
open science

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

Érik Martin-Dorel, Sergei Soloviev

► **To cite this version:**

Érik Martin-Dorel, Sergei Soloviev. A Formal Study of Boolean Games with Random Formulas as Payoff Functions. 22nd International Conference on Types for Proofs and Programs (TYPES 2016), May 2016, Novi Sad, Serbia. pp.1-22, 10.4230/LIPIcs.TYPES.2016.14 . hal-02651342

HAL Id: hal-02651342

<https://hal.science/hal-02651342>

Submitted on 29 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License



Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: <https://oatao.univ-toulouse.fr/22314>

To cite this version:


Martin-Dorel, Erik and Soloviev, Sergei *A Formal Study of Boolean Games with Random Formulas as Payoff Functions*. (2018) In: 22nd International Conference on Types for Proofs and Programs (TYPES 2016), 23 May 2016 - 26 May 2016 (Novi Sad, Serbia).

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

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

Érik Martin-Dorel

Lab. IRIT, Univ. of Toulouse, CNRS, IRIT Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex 9, France
erik.martin-dorel@irit.fr

 <https://orcid.org/0000-0001-9716-9491>

Sergei Soloviev¹

Lab. IRIT, Univ. of Toulouse, CNRS, IRIT Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex 9, France
sergei.soloviev@irit.fr

Abstract

In this paper, we present a probabilistic analysis of Boolean games. We consider the class of Boolean games where payoff 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.

2012 ACM Subject Classification Theory of computation → Higher order logic, Theory of computation → Algorithmic game theory, Mathematics of computing → Stochastic processes

Keywords and phrases Boolean games, Random process, Coq formal proofs

Digital Object Identifier 10.4230/LIPIcs.TYPES.2016.14

Supplement Material <https://doi.org/10.5281/zenodo.1317609>

Funding This work was partly supported by the FAGames project of LabEx CIMI.

Acknowledgements The authors would like to thank Evgeny Dantsin and Jan-Georg Smaus for fruitful discussions on algorithmic games and other topics, as well as the anonymous referees for their remarks and feedback on our article. The first author also wishes to thank Cyril Cohen and Enrico Tassi for advice and enhancement suggestions for our **under** tactic.

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

¹ S. Soloviev was partly supported by the Government of Russian Federation Grant 08-08 at ITMO University, Saint-Petersburg, Russia (associated researcher).

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 paper 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 payoff 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 made 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 basic properties of Boolean functions.

(i) The domain of these functions is $\mathbf{2}^n = \{\text{false}, \text{true}\}^n$, 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 form² that is satisfied by exactly these vectors (to each vector corresponds the conjunction of variables and their negations: to true at the i -th place corresponds v_i and to false 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, $\mathbb{P}(\text{true})$ 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:

► **Definition 1.** Let ω_1, ω_2 be two Boolean functions ($\omega_1, \omega_2 \in \Omega := \mathbf{2}^{\mathbf{2}^n}$). We shall write $\omega_1 \Rightarrow_0 \omega_2$ and say that “ ω_2 is true on ω_1 ” if for each vector $v \in \mathbf{2}^n$, $\omega_1(v) = \text{true}$ implies $\omega_2(v) = \text{true}$. Also, this is equivalent to the inclusion of ω_1 in ω_2 , seen as subsets of $\mathbf{2}^n$:

$$\forall \omega_1, \omega_2 \in \mathbf{2}^{\mathbf{2}^n}. (\omega_1 \Rightarrow_0 \omega_2) \iff (\omega_1 \subseteq \omega_2).$$

² In particular, the empty subset may be identified with the empty DNF, that is the constant false (the neutral element of the disjunction).

► **Remark.** We may see random Boolean functions as (not necessarily independent) vectors of 2^n random variables with values in $\mathbf{2} = \{\text{false}, \text{true}\}$.

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

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

Regarding the probability distributions on the spaces of Boolean formulas: as it is noticed in [8], it is often assumed that all Boolean functions on a given number of variables have the same probability (see also [16]). In this paper 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 functions constructed through a finite Bernoulli process, specialise our results in this simpler setting, then discuss the relevance of these results. 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 growth rate of the aforementioned probability, with respect to the knowledge of the second player’s choices. Section 6 is devoted to technical remarks about the formalisation of our results within the Coq [5] formal proof assistant. Finally, a discussion on the notion of “non-guaranteed win” and its relationship with the order of moves is presented in Appendix A.

All the results of the paper have been formally verified within Coq.³

The Coq code is available online at <https://github.com/erikmd/coq-bool-games> and it also has been archived, see [14].

Beyond the fact that formal certification is interesting in itself to the TYPES community, it is nowadays common in the development and characterisation 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 $\mathcal{S} = \mathbf{2}^{2^{2^n}}$, 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 2** (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 payoff function of A , and its negation as the payoff

³ In the sequel of the paper, all definitions and theorems will be stated in mathematical syntax, and the corresponding Coq identifier will be given between brackets.

function for B . We shall assume that A controls the first k variables, and B the remaining $n - k$ variables.

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 payoff function $F : \mathbf{2}^n \rightarrow \mathbf{2}$, which can thereby be viewed as a function $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$ (mapping a profile strategy to a Boolean outcome). In the sequel of the paper, we shall identify these two possible types for the function F – while in the formal development they will be encoded respectively as `(bool_fun n)` and `(bool_game n k)`.

► **Definition 3** (`winA`). For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$, a strategy $a = (a_1, \dots, a_k)$ of player A is winning if it wins against any strategy $b \in \mathbf{2}^{n-k}$ of B :

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

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

In other words, a is winning if the payoff function is equal to `true` on all vectors of length n that “extend” a . This led us to introduce the following

► **Definition 4** (`w_`, `W_`). For any $a \in \mathbf{2}^k$, let ω_a be the set of vectors in $\mathbf{2}^n$ that extend a :

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

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

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

Lemma 5 implies that the probability that a winning strategy exists satisfies:

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

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

► **Theorem 6** (`Pr_bigcup_incl_excl`). For any finite probability space $(\Omega, \mathcal{S}, \mathbb{P})$ and any sequence of events $(S_i)_{0 \leq i < n}$, we have:

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

Proof. For proving this theorem in Coq we formalise a small theory of indicator functions $\text{Ind}_S : \Omega \rightarrow \{0, 1\}$ for any finite set $S \subseteq \Omega$, including the fact that the expectation satisfies

$\mathbb{E}(\text{Ind}_S) = \mathbb{P}(S)$, then formalise an algebraic proof of the inclusion-exclusion formula.⁴ 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. ◀

Hence the following result:

► **Theorem 7** (`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:*

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

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 7 is applicable with *any* probability \mathbb{P} , but it is not easy to handle. In the upcoming section, we shall investigate in more detail the case when \mathbb{P} is relatively simple.

Before refining Theorem 7 with specific definitions of \mathbb{P} , we formally study the dual case (i.e., the existence of a winning strategy from the point of view of player B).

► **Definition 8** (`winB`). For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$, a strategy $b = (b_1, \dots, b_{n-k})$ of player B is winning if it wins against any strategy $a \in \mathbf{2}^k$ of A :

$$\text{win}_B[F](b) := \forall a \in \mathbf{2}^k. F(a, b) = \text{false}.$$

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

A first result consists in showing that player B wins in a given game if and only if player A wins in the “dual game”.

► **Lemma 9** (`winB_eq`). *Any Boolean game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$ (with n variables, k of which are controlled by player A) can be associated with a dual Boolean game $F' : \mathbf{2}^{n-k} \times \mathbf{2}^k \rightarrow \mathbf{2}$ such that*

$$\text{win}_B[F](b) \iff \text{win}_A[F'](b)$$

Proof. First, we define the dual game $F' := \text{bool_game_sym}(F)$ associated with F as:

$$F' := (b, a) \mapsto \neg F(a, b).$$

Then, we define `bool_game_sym'` (the inverse of function `bool_game_sym`) and show that both functions are bijections. In the formal development, the related lemmas are named `bool_game_sym_bij` and `bool_game_sym'_bij`. ◀

We then deduce the following result, which relates the probability of existence of a winning strategy for player B with respect to that of player A .

⁴ taking inspiration from the proof path presented at https://en.wikipedia.org/wiki/Inclusion-exclusion_principle#Algebraic_proof

► **Theorem 10** (`Pr_ex_winB`). For any finite probability space $(\Omega, \mathcal{S}, \mathbb{P})$, the probability that there exists some strategy $b = (b_1, \dots, b_{n-k})$ of B that is winning satisfies:

$$\mathbb{P}(\exists b : \mathbf{2}^{n-k}. \text{win}_B[F](b)) = \mathbb{P}(\exists a : \mathbf{2}^{n-k}. \text{win}_A[F'](a)),$$

where $F' := \text{bool_game_sym}(F)$.

Proof. The proof straightforwardly derives from Lemma 9. ◀

Finally, we prove the intuitive fact that the events “ $\exists a. \text{win}_A(a)$ ” and “ $\exists b. \text{win}_B(b)$ ” are disjoint, and thereby their probability adds up:

► **Lemma 11** (`Pr_ex_winA_winB_disj`). For any finite probability space $(\Omega, \mathcal{S}, \mathbb{P})$, we have:

$$\mathbb{P}(\exists a. \text{win}_A(a) \vee \exists b. \text{win}_B(b)) = \mathbb{P}(\exists a. \text{win}_A(a)) + \mathbb{P}(\exists b. \text{win}_B(b)).$$

Proof. Given the definitions of win_A and win_B , for a given game F and any strategies a and b , the events “ $\text{win}_A(a)$ ” and “ $\text{win}_B(b)$ ” are disjoint. So the proof path just amounts to lift this fact (considering existence) and use the additivity of \mathbb{P} . ◀

3 Bernoulli Process and Winning Strategies

In this section we still consider the space $\Omega = \mathbf{2}^{2^n}$ of random Boolean formulas of n variables endowed with the discrete σ -algebra $\mathcal{S} = \mathbf{2}^{2^{2^n}}$, 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 write $q = 1 - p$.

In the sequel, we shall often identify Boolean functions $F : \mathbf{2}^n \rightarrow \mathbf{2}$ and their truth-set $F^{-1}(\{\text{true}\}) \in \mathbf{2}^{2^n}$. 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}
```

and the inverse of this function is formalised as a function `DNF_of` (disjunctive normal form).

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 the truth-set of F or not. We obtain the following result:

► **Lemma 12** (`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:

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

where m denotes the number of vectors in the truth-set of F , and $2^n - m$ 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 true, 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 13** (`Pr_implies0_Bern`). *Let $S \subseteq \mathbf{2}^n$, and let us write $m := \text{Card } S$. Then the probability that F is true on S satisfies: $\mathbb{P}_{n;p}(S \Rightarrow_0 F) = p^m$.*

Proof. We follow the following proof path:

$$\begin{aligned}
\mathbb{P}_{n;p}(S \Rightarrow_0 F) &= \sum_{\substack{F \\ S \subseteq F}} \mathbb{P}_{n;p}(\{F\}) \\
&= \sum_{\substack{S' \subseteq \mathbf{2}^n \setminus S \\ F = S \cup S'}} \mathbb{P}_{n;p}(\{F\}) \\
&= \sum_{S' \subseteq \mathbf{2}^n \setminus S} p^{\text{Card}(S \cup S')} q^{2^n - \text{Card}(S \cup S')} \text{ by Lemma 12} \\
&= \sum_{m'=0}^{2^n - m} \binom{2^n - m}{m'} p^{m+m'} q^{2^n - m - m'} \\
&= p^m \sum_{m'=0}^{2^n - m} \binom{2^n - m}{m'} p^{m'} q^{(2^n - m) - m'} \\
&= p^m (p + q)^{2^n - m} \\
&= p^m. \quad \blacktriangleleft
\end{aligned}$$

► **Lemma 14** (`card_w_a_Bern`). *For any strategy a of player A , we have*

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

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

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

Proof. This result is an immediate consequence of Lemmas 13 and 14. ◀

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

► **Lemma 16** (`w_trivIset`). *The truth-sets of w_a (for $a \in J \subset \mathbf{2}^k$) are pairwise disjoint.*

Proof. By contradiction: if we had $a, a' \in \mathbf{2}^k$ such that $w_a \neq w_{a'}$ and $w_a \cap w_{a'} \neq \emptyset$, then let us pose $x \in w_a \cap w_{a'}$. By unfolding Definition 4, this means that the first k bits of x coincides with all bits of a , and likewise for a' . This implies that $a = a'$ and thereby $w_a = w_{a'}$, which contradicts the initial hypothesis. ◀

Lemma 16 implies that we have

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

We can now prove the following

► **Theorem 17 (Pr_ex_winA_Bern).** *For any n and k , if $\mathbb{P}_{n;p}$ follows the Bernoulli scheme that we previously constructed, the probability that player A has a winning strategy is:*

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

Proof. Thanks to Theorem 7, we can write:

$$\begin{aligned} \mathbb{P}_{n;p}(\exists a : \mathbf{2}^k. \text{win}_A(a)) &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \text{Card } J=m}} \mathbb{P}_{n;p} \left(\bigcap_{a \in J} W_a \right) \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \text{Card } J=m}} \mathbb{P}_{n;p} \left(\bigcap_{a \in J} [w_a \Rightarrow_0 F] \right) \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \text{Card } J=m}} \mathbb{P}_{n;p} \left[\left(\bigcup_{a \in J} w_a \right) \Rightarrow_0 F \right] \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \text{Card } J=m}} p^{(\text{Card}(\bigcup_{a \in J} w_a))} \text{ by Lemma 13} \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \sum_{\substack{J \subseteq \mathbf{2}^k \\ \text{Card } J=m}} p^{m \cdot 2^{n-k}} \text{ by using (3) and Lemma 14} \\ &= \sum_{m=1}^{2^k} (-1)^{m-1} \binom{2^k}{m} p^{m \cdot 2^{n-k}} \\ &= 1 - \sum_{m=0}^{2^k} \binom{2^k}{m} (-p^{2^{n-k}})^m 1^{2^k-m} \\ &= 1 - \left(1 - p^{2^{n-k}}\right)^{2^k}. \quad \blacktriangleleft \end{aligned}$$

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

► **Corollary 18 (Pr_ex_winB_Bern).** *For any p , n , k , if $\mathbb{P}_{n;p}$ denotes the considered Bernoulli scheme (with parameters $0 \leq p \leq 1$ 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:*

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

Proof. The result follows from Theorems 10 and 17. Also, the proof makes use of our **under** tactic for rewriting under lambdas (it will be presented in Section 6). ◀

Table 1 The probability that a winning strategy exists neither for A nor for B ($n = 10$).

p\k	1	2	3	4	5	6	7	8	9
0.25	1.52e-184	5.11e-43	1.37e-6	0.525	0.997	1	0.998	0.367	4.46e-15
0.5	1.07e-64	6.68e-8	0.606	0.999	1	0.999	0.606	6.68e-8	1.07e-64
0.75	4.46e-15	0.367	0.998	1	0.997	0.525	1.37e-6	5.11e-43	1.52e-184

Table 2 The probability that a winning strategy exists neither for A nor for B ($n = 20$).

p\k	1	2	3	4	5	6	7	8	9	10
0.25	1.27e-188231	2.04e-43307	2.15e-6005	1.99e-287	3.72e-2	1	1	1	1	1
0.5	1.32e-65504	2.74e-7348	1.61e-223	0.368	1	1	1	1	1	1
0.75	7.53e-14696	2.58e-446	0.135	1	1	1	1	1	1	1

p\k	11	12	13	14	15	16	17	18	19
0.25	1	1	1	1	1	1	0.135	2.58e-446	7.53e-14696
0.5	1	1	1	1	1	0.368	1.61e-223	2.74e-7348	1.32e-65504
0.75	1	1	1	1	3.72e-2	1.99e-287	2.15e-6005	2.04e-43307	1.27e-188231

► **Corollary 19** (`Pr_nex_winA_winB_Bern`). For any p, n, k , if $\mathbb{P}_{n;p}$ denotes the considered Bernoulli scheme (with parameters $0 \leq p \leq 1$ and $n \in \mathbb{N}$) and if k denotes the number of variables controlled by player A , then the probability that no player has a winning strategy is:

$$\mathbb{P}_{n;p} \left(\neg \left((\exists a. \text{win}_A(a)) \vee (\exists b. \text{win}_B(b)) \right) \right) = \left(1 - p^{2^{n-k}} \right)^{2^k} + \left(1 - (1-p)^{2^k} \right)^{2^{n-k}} - 1. \quad (4)$$

Proof. The result follows from Lemma 11, Theorem 17 and Corollary 18. ◀

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 $p \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$, $n \in \{10, 20\}$, $0 < k < n$, the probability that a winning strategy exists neither for A nor for B (cf. Equation (4)) is given in Tables 1 and 2 (the values were computed using Sollya⁵ with 3-digit decimal output). In both tables, it should be noted that 1 actually means a value extremely close to 1, not 1 exactly.

Also, one may notice that when $p \in (0, 1)$ is fixed, $k = c \cdot n$ for a given constant $0 < c < 1$, and n tending towards $+\infty$, the probability that a winning strategy exists neither for player A , nor for player B , tends to 1.

If (for some game F) a winning strategy exists neither for A nor 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 $F(a, b) = \text{false}$. If B makes his choice after A , he may always win. Similarly, if A makes his choice after B , he may always win.

We shall elaborate on this observation and give a motivating example in Appendix A.

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.

⁵ <http://sollya.gforge.inria.fr/>

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 20** (`winA_knowing`). For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$ 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 $b_{1:s}$:

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

where

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

For relating this predicate with that of Definition 3, the proof of the following lemma is immediate:

► **Lemma 21** (`winA_knowingE`). For any game $F : \mathbf{2}^k \times \mathbf{2}^{n-k} \rightarrow \mathbf{2}$ and any bit-vectors $b_{1:s} \in \mathbf{2}^s$ and $a \in \mathbf{2}^k$, we have:

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

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

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

Now, to compute the probability $\mathbb{P}_{n;p}(\forall b_{1:s} \in \mathbf{2}^s. \exists a \in \mathbf{2}^k. \text{win}_A(a \mid b_{1:s}))$ in the space $(\Omega, \mathcal{S}, \mathbb{P}_{n;p})$ introduced in Section 3, we shall first construct a probability space $(\Omega', \mathcal{S}', \mathbb{P}')$ that is provably isomorphic to $(\Omega, \mathcal{S}, \mathbb{P}_{n;p})$, but which is simpler to handle.

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

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

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 correspondence between $\Omega_{b_{1:s}}$ and $\mathbf{2}^{2^{n-s}}$. We shall denote the corresponding bijections by $g : \Omega_{b_{1:s}} \rightarrow \mathbf{2}^{2^{n-s}}$ and $h : \mathbf{2}^{2^{n-s}} \rightarrow \Omega_{b_{1:s}}$. 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 correspondence between Ω' and $\Omega = \mathbf{2}^{2^n}$. We shall denote the corresponding bijections by $g' : \Omega' \rightarrow \Omega$ and $h' : \Omega \rightarrow \Omega'$. 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 22** (`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 23** (`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 $(\pi_i : \Omega_{\Pi} \rightarrow \Omega_i)_{i \in I}$ 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 24** (`Pr_ex_winA_knowing_Bern`). *For all $p \in [0, 1]$ and for all integers n, k, s satisfying $0 \leq s \leq n - k \leq n$, 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:*

$$\mathbb{P}_{n;p} (\forall b_{1:s} \in \mathbf{2}^s. \exists a \in \mathbf{2}^k. \text{win}_A(a \mid b_{1:s})) = \left(1 - \left(1 - p^{2^{n-k-s}} \right)^{2^k} \right)^{2^s}. \quad (5)$$

Proof. We follow the following proof path:

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

hence by using Lemma 21

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

hence by using Lemma 22

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

hence by using elementary facts on g , g' and the bgk function defined in Lemma 21

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

hence by using Lemma 23

$$= \prod_{b_{1:s} \in \mathbf{2}^s} \mathbb{P}_{b_{1:s}} \{S \in \Omega_{b_{1:s}} \mid \exists a \in \mathbf{2}^k. \text{win}_A[g(S)](a)\}$$

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

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

hence by definition of g and h

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

hence by using Theorem 17 in the case of random Boolean functions with $n - s$ variables

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

We may compare this probability with the probability of existence of unconditionally winning strategy studied in Section 3 (Theorem 17). The upcoming section will focus on this question.

► **Remark.** In Theorems 17 and 24, we formally studied the *probability of guaranteed win* (knowing partial information on the opponent), that is, the probability that for every value taken by the first s variables of B ,⁶ there exists a strategy for A that wins against all strategies of B given this fixed value of the first s variables of B . This problem is purely combinatorial and does not depend on the “preferences” of B (regarding the variables that he controls). So this probability will typically be different from the probability of non-guaranteed win for player A , as this latter probability could be influenced by the preferences of B for some choices, the dependency of these choices on F , and so on.

⁶ Theorem 17 being a particular case of Theorem 24 ($s = 0$).

5 Probability of Guaranteed Win: Growth Rate

Using the result given by Theorem 24, we would like to study how the probability of guaranteed win grows with each bit of information concerning the choice of B .

For fixed values of $p \in (0, 1)$, $n, k \in \mathbb{N}$ such that $0 < k < n$, and for $0 \leq s \leq n - k$, let us write $g(s)$ the quantity given in Equation (5).

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

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

Then, an interesting question may be: what is the order of growth of the difference

$$\phi(s) := g(s) - g(0) \quad (\in [0, 1])$$

with respect to s ? The following result is a first answer to this question:

► **Theorem 25 (ϕ _ineq).** *For any $p \in (0, 1)$, $n, k \in \mathbb{N}^*$ such that $0 \leq s \leq n - k$, if the following condition holds:*

$$2^k p^{2^{n-k-s}} < 1, \tag{6}$$

then we have

$$\phi(s) > \left(2^{(k-1)2^s} - 2^k\right) p^{2^{n-k}}, \tag{7}$$

where

$$\phi(s) = g(s) - g(0) = \left(1 - \left(1 - p^{2^{n-k-s}}\right)^{2^k}\right)^{2^s} - \left(1 - \left(1 - p^{2^{n-k}}\right)^{2^k}\right).$$

In particular, condition (6) is satisfied as soon as the following, stronger condition is satisfied:

$$s \leq (n - k) - \log_2(k + 1) + \log_2(|\log_2 p|). \tag{8}$$

Proof. Let us write $t = p^{2^{n-k-s}}$. By the binomial formula, we have:

$$1 - (1-t)^{2^k} = 2^k t - (2^k t)^2 \sum_{i=2}^{2^k} (-1)^i 2^{-2k} \binom{2^k}{i} t^{i-2}. \tag{9}$$

We notice that if (6) holds, that is if $2^k t < 1$, then the absolute value of the $(i + 1)$ th member of the sum \sum in (9) 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 (6) holds, then the sum (positive) is less than or equal to its first term. Moreover, the first term of the sum \sum in (9) is $2^{-2k} \frac{2^k(2^k-1)}{2} < \frac{1}{2}$. So, if (6) is satisfied for some n, k, s , then from (9) we obtain

$$1 - (1-t)^{2^k} \geq 2^k t - \frac{1}{2}(2^k t)^2 = 2^k t \left(1 - \frac{1}{2} 2^k t\right) > 2^{k-1} t. \tag{10}$$

Thus, under these conditions

$$g(s) = \left(1 - \left(1 - p^{2^{n-k-s}}\right)^{2^k}\right)^{2^s} > 2^{(k-1)2^s} p^{2^{n-k}}. \tag{11}$$

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

$$g(0) = \left(1 - (1 - p^{2^{n-k}})^{2^k}\right) = 2^k p^{2^{n-k}} - \sum_{i=1}^{2^k} \binom{2^k}{i} (-p^{2^{n-k}})^i \leq 2^k p^{2^{n-k}} \quad (12)$$

Combining (11) and (12) yields the following inequality:

$$g(s) - g(0) > 2^{(k-1)2^s} p^{2^{n-k}} - 2^k p^{2^{n-k}} = (2^{(k-1)2^s} - 2^k) p^{2^{n-k}}. \quad (13)$$

Finally, the following condition is obviously stronger than (6):

$$2^k p^{2^{n-k-s}} \leq \frac{1}{2},$$

which 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

$$s \leq (n-k) - \log_2(k+1) + \log_2(|\log_2 p|),$$

which is thereby a sufficient condition for (6). ◀

For example, if $p = \frac{1}{2}$, condition (8) becomes

$$s \leq (n-k) - \log_2(k+1). \quad (14)$$

And if $0 < p < \frac{1}{2}$, $\log_2(|\log_2 p|) > 0$ so we have $-\log_2(k+1) < -\log_2(k+1) + \log_2(|\log_2 p|)$, and thereby we can also rely on condition (14).

It can be noted that inequality (7) essentially gives an order of growth of $2^{(k-1)2^s}$ with respect to the quantity of information s (number of extra bits known by player A), which is much faster than usual orders of growth of s or 2^s .

Still, a more refined study of the behaviour of the function that describes the growth of $g(s)$ (the probability of a guaranteed win depending on s) requires much more effort and space that we could give to it in this exploratory paper. For example, it is intuitively clear that the graph of this function is a typical ‘‘S-form’’ curve (see Figure 1), but it is not easy to determine where the critical points are placed; and for small values of the parameter s , the inequality we get in (7) may be too rude to place with sufficient precision. However the behaviour of this function g may be of interest for strategic planning concerning both players. This remains subject of future work.

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.

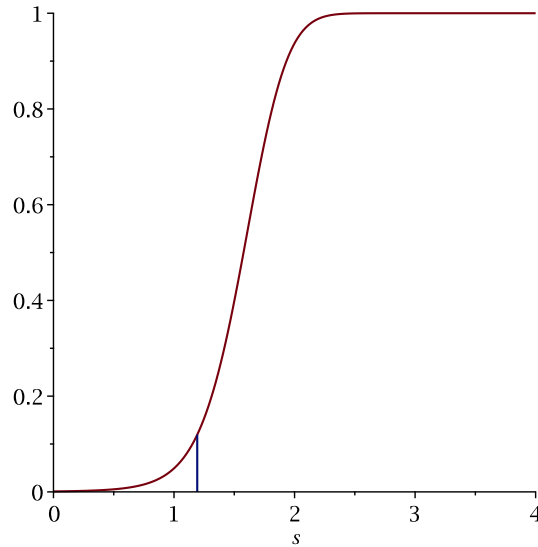


Figure 1 Graph of $g(s)$, for the parameters $p = \frac{1}{2}$, $n = 10$, and $k = 6$. The vertical line at $s \approx 1.19$ indicates the largest $s \in \mathbb{R}$ that satisfies (8).

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, 15] 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 library⁷ 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 library⁸ 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 library⁹.

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).

⁷ <https://www.lri.fr/~paulin/ALEA/>

⁸ <https://staff.aist.go.jp/reynald.affeldt/shannon>

⁹ <https://math-comp.github.io/math-comp/>

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, the sum

$$\sum_{\substack{i=1 \\ i \text{ odd}}}^4 i^2$$

can be formally written as `\sum_(1 <= i < 5 | odd i) i^2`, which amounts to the following term if we get rid of the `\sum` notation:

```
bigop 0 (index_iota 1 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 Coq tactic “`under`” for *rewriting under the lambdas* of big-operators.

A generalised version of our tactic, also applicable for MathComp notions such as matrices, polynomials, and so on, is available online at <https://github.com/erikmd/ssr-under-tac> and we plan to submit it for possible inclusion in MathComp.

Below is a typical example of use for that generalised implementation of the `under` tactic.

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
```

Dependent product of finTypes

MathComp has built-in support for finite functions: for any $(A:\text{finType})$ and $(T:\text{Type})$, the notation $\{\text{ffun } A \rightarrow T\}$ 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 $\{\text{ffun } A \rightarrow T\}$ 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 6; (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 23.

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 subsumes the simpler case where all Boolean functions have the same probability: this latter case corresponds to choosing $p = \frac{1}{2}$ in our setting).

In this paper 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 where s is the quantity of information (number of extra bits) known by 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].

References

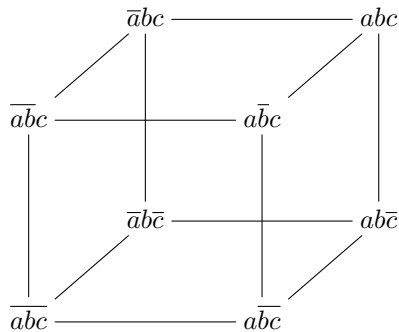
- 1 Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues. Formalization of Shannon’s theorems. *J. Autom. Reasoning*, 53(1):63–103, 2014. doi:10.1007/s10817-013-9298-1.
- 2 Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Sci. Comput. Program.*, 74(8):568–589, 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, 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 8.8*, 2018. URL: <https://coq.inria.fr/distrib/V8.8.0/refman/>.
- 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. doi:10.1007/978-3-642-39634-2_14.
- 10 Paul Harrenstein. *Logic in Conflict. Logical Explorations in Strategic Equilibrium*. PhD thesis, Utrecht University, 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 Erik Martin-Dorel and Sergei Soloviev. erikmd/coq-bool-games: BoolGames, 2018. doi:10.5281/zenodo.1317609.
- 15 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.
- 16 John Riordan and C. E. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21:83–93, 1942.

A Non-Guaranteed Win: When the Order of Choices Matters

Let us consider an example of three variables a, b, c and two players, Alice who controls a and Bob who controls b, c . Let us consider all possible Boolean functions as payoff functions. There are 256 that may be identified with the subsets of the nodes of the cube below. Each subset is interpreted as the disjunction of the conjunctions in the nodes.



It makes sense to analyse this situation in a purely *combinatorial* way before we consider *randomly generated payoff functions*. We notice the following facts:

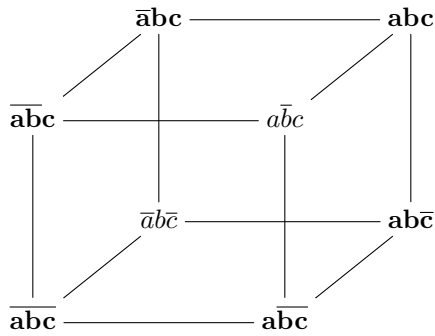
- Alice has an unconditionally winning strategy in 31 cases (these cases correspond to all subsets that contain all nodes of either the face with a or the face with \bar{a} ; the number of subsets is easily counted by the formula of inclusions-exclusions).
- Bob has an unconditionally winning strategy in 175 cases (the cases correspond to the subsets that *do not* intersect with one of the four edges defined by the choice of two literals among b, c, \bar{b}, \bar{c} ; the number is counted as above).
- There are 50 cases when neither Alice nor Bob has an unconditionally winning strategy. In these cases the order of choice matters:
 - If Alice chooses the value of a first, then Bob has a winning strategy (he may win in all these cases).
 - Similarly, if Bob chooses the values of b, c first then Alice may win in all these cases.

Now let us consider in more detail the case where the order of choices is $B - A - B$. In fact, here we need to distinguish three subcases:

1. Bob may give a value to any of b, c at his first step.
2. At his first step, Bob gives a value to b , and at his second to c .
3. At his first step, Bob gives a value to c , and at his second to b .

It can be noted that these three variants may correspond to preferences or to obligations (extra constraints) concerning Bob, in line with the remark at the end of Section 4 (page 12). We elaborate on these three subcases below.

1. The choice of Bob may be interpreted as the selection of one of the four faces of the cube that corresponds to b, \bar{b}, c, \bar{c} respectively. There are four cases when Alice may win if she knows the first choice of Bob. One subset is shown below in bold, other are obtained by rotation. (We exclude the cases of unconditional win that were counted before.)



In the case displayed above, if Bob has chosen $b = \text{true}$ then Alice has to choose $a = \text{true}$ and wins because the remaining formula will be $c \vee \bar{c}$.

2. If at the first step Bob must choose the value of b , then it may be seen as the choice of one of the *two* faces of the cube that correspond to b or \bar{b} . This gives Alice more possibilities to win. Indeed, she may win if the subset of nodes includes either $\bar{a}\bar{b}c, \bar{a}bc, ab\bar{c}, abc$ or $a\bar{b}\bar{c}, a\bar{b}c, \bar{a}b\bar{c}, \bar{a}bc$. We may add one or more nodes to each subset of four, but if we exclude the previously considered cases, we shall have 12 more cases when Alice may win.
3. Similar analysis shows that it will be 12 cases (not considered previously) where Alice may win if Bob must choose the value of c first.

It is important to notice that the choices of Alice and Bob are not necessarily interpreted as the choices of logical values of a, b, c . This model may be used to model any binary choice. Indeed, let $a = \text{true}$ mean the choice of some value v_a and $a = \text{false}$ mean the choice of v'_a by Alice. Similarly, Bob may choose one of v_b, v'_b and one of v_c, v'_c . Instead of conjunction of literals (e.g., $\bar{a}\bar{b}c$) let us take for each such conjunction a predicate¹⁰ $P_{\bar{a}\bar{b}c}(x, y, z)$ which is true if and only if $x = v_a, y = v'_b, z = v'_c$. Instead of considering the disjunction of these conjunctions, let us take the disjunction of corresponding predicates. It appears that the logical value of the result will exactly be the logical value of the payoff function represented by the DNF (or Boolean function).

The roles of Alice and Bob may be seen as the roles of “coaches” who choose the players for a series of matches. Alice wins if her “champion” wins at least one match. Also, to come back to the situation with random payoff functions, it makes perfect sense that in a real tournament the coach cannot know in advance which matches will be necessary to play.

¹⁰ defined for $(x, y, z) \in \{v_a, v'_a\} \times \{v_b, v'_b\} \times \{v_c, v'_c\}$

The same idea may be used to model markets (the choice $a = \text{true}$ may mean that Alice orders to buy a certain product a , $a = \text{false}$ that she orders to sell, and the presence of \overline{abc} that she makes profit when she buys at the same time when Bob sells his two products).

This analysis also clearly shows what may be the role of introduction of random choice of payoff functions. It takes into account certain amount of unpredictability in a real situation. Notice that it does not eliminate some “geometric flavour” displayed in the above example.

However, as we emphasised before, we intend to use probability mostly for the analysis of the totality of games with all possible Boolean functions as payoff, rather than for considering one game with a randomly-chosen payoff function (though this may sometimes make sense).

The choice of probability distribution will influence the relative “weight” of the cases that we considered above in a purely combinatorial way, and has to be taken into account when additional conditions are considered, such as the order of moves or access to the information.

For example, if the probability parameter p takes a value of $\frac{1}{2}$ (i.e., if we focus on the instance $\mathbb{P}_{3; \frac{1}{2}}$ of the Bernoulli process presented in Section 3), this will give the uniform distribution on the 256 cases considered in the appendix.