Exploring the Average Values of Boolean Functions via Asymptotics and Experimentation

Robin Pemantle*
Department of Mathematics
University of Pennsylvania
Philadelphia, PA 19104–6395
pemantle@math.upenn.edu

Mark Daniel Ward
Department of Mathematics
University of Pennsylvania
Philadelphia, PA 19104–6395
ward2@math.upenn.edu

Abstract

In recent years, there has been a great interest in studying Boolean functions by studying their analogous Boolean trees (with internal nodes labeled by Boolean gates; leaves viewed as inputs to the Boolean function). Many of these investigations consider Boolean functions of n variables and m leaves. Our study is related but has a quite different flavor.

We investigate the mean output X_n of a Boolean function defined by a complete Boolean tree of depth n. Each internal node of such a tree is labeled with a Boolean gate, via $2^n - 1$ IID fair coin flips. The value of the input at each leaf can be simply fixed at 1/2, so the randomness of X_n derives only from the selection of the gates at the internal nodes.

For each n, there are $2^{(2^n-1)}$ possible Boolean binary trees to consider, so we cannot expect to obtain a complete description of the probability distribution of X_n for large n. Therefore, we perform a twofold investigation of the X_n, using both asymptotics and experiments. We prove that, with probability 1, $X_n \to 0$ or $X_n \to 1$. Then we directly compute the asymptotics of the first four moments of X_n. Writing $Z_n = X_n(1 - X_n)$, we also prove that $E(Z_n)$ and $E(Z_n^2)$ are both $\Theta(1/n)$. Finally, we utilize C++ and a significant amount of computation and experimentation to obtain a more descriptive understanding of X_n for small values of n (say, $n \leq 100$).

1 Introduction.

We first outline the construction of a Boolean function using a binary tree. We utilize complete binary trees T_n of depth n. At each of the internal nodes, we place either an AND gate or an OR gate, with probability 1/2 each. Selection of the gates at distinct nodes is independent, so the gates are essentially chosen by IID fair coin flips. In other words, we uniformly select a vector consisting of $2^n - 1$ AND's and OR's, namely $\vec{g}_n \in \{\text{AND}, \text{OR}\}^{2^n-1}$. By labeling the internal nodes of a complete binary tree of depth n with this collection $\vec{g}_n$ of $2^n - 1$ gates, we naturally define a random Boolean function $\phi_n(\vec{g}_n) : \{0,1\}^{2^n} \to \{0,1\}$. The leaves of the tree, say $i_1, i_2, \ldots, i_{2^n}$, are considered as the inputs to the Boolean function. The output at the root of the tree is viewed as the output of the Boolean function. Thus we write

$$\phi_n(\vec{g}_n)(i_1, i_2, \ldots, i_{2^n}) \in \{0, 1\}$$

for each $(2^n - 1)$-tuple $\vec{g}_n$ of gates and each 2^n-tuple of inputs $i_1, i_2, \ldots, i_{2^n}$.

In this investigation, we are interested in studying the behavior of the random variable X_n, which denotes the *mean* output of $\phi_n(\vec{g}_n)$ on 2^n Boolean inputs. In other words,

$$X_n := \frac{1}{2^{2^n}} \sum_{i_1, i_2, \ldots, i_{2^n}} \phi_n(\vec{g}_n)(i_1, i_2, \ldots, i_{2^n})$$

We observe that X_n is a random variable because the selection of the $2^n - 1$ gates in $\vec{g}_n$ is performed at random. Once the selection of the gates $\vec{g}_n$ is determined, then X_n is completely determined, because X_n is the average of all possible 2^{2^n} selections of inputs $i_1, i_2, \ldots, i_{2^n}$ to the Boolean tree described above. So the randomness of X_n does not stem from a random choice of the inputs $i_1, i_2, \ldots, i_{2^n}$ at all; X_n's randomness only depends on the random selection of gates at the internal nodes of the tree. Once the gates at the nodes are chosen, then we average over all possible inputs to the binary tree.

For each selection $\vec{g}_n$ of gates, we note that $\phi_n(\vec{g}_n)$ is a function with 2^n inputs. If the inputs $i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_{2^n}$ are all fixed, then $\phi_n(\vec{g}_n)$ is a *linear* function of i_j. Since $i_j \in \{0, 1\}$ for each j, then we conclude that X_n can be computed easily, once the gates $\vec{g}_n$ are chosen, by simply taking 1/2 as the value of each input i_j to the Boolean function $\phi_n(\vec{g}_n)$. In other words, for each selection of $\vec{g}_n$, we have

$$X_n = \phi_n(\vec{g}_n)(1/2, 1/2, \ldots, 1/2);$$

in this representation, it is perhaps easiest to see that the randomness of X_n is due to the random selection of the gates in the $(2^n - 1)$-tuple $\vec{g}_n$.

*Supported by NSF grant DMS-0401246.

An example is useful for clarification. Consider the selection of $\vec{g}_3$ given below in this tree of depth 3:

$$\begin{array}{cccccccccccccc} & & & & & & & \vee & & & & & & \\ & & & \vee & & & & & & & & \wedge & & \\ & \wedge & & & & \vee & & & & \vee & & & \wedge & \\ i_1 & & i_2 & & i_3 & & i_4 & & i_5 & & i_6 & i_7 & & i_8 \end{array}$$

For complete trees of depth 3, we see that X_3 denotes the mean output of a Boolean random function with gates $\vec{g}_3$. If the choice of $\vec{g}_3$ is the one given above, this results in X_3 having the value 217/256. To see this, simply evaluate the tree

$$\begin{array}{cccccccccccccc} & & & & & & & \vee & & & & & & \\ & & & \vee & & & & & & & & \wedge & & \\ & \wedge & & & & \vee & & & & \vee & & & \wedge & \\ \frac{1}{2} & & \frac{1}{2} & & \frac{1}{2} & & \frac{1}{2} & & \frac{1}{2} & & \frac{1}{2} & \frac{1}{2} & & \frac{1}{2} \end{array}$$

Evaluating such a tree with inputs besides the familiar $\{0, 1\}$ requires a bit of explanation. The evaluation of expressions such as $i_1 \wedge i_2$ is quite easy. This expression, for instance, evaluates to 1 if both i_1 and i_2 have the value 1; otherwise, the expression evaluates to 0. Unfortunately, this evaluation is useful only for $i_1, i_2 \in \{0, 1\}$. So we instead use the following equivalent interpretation (which is quite standard). We write

$$\begin{aligned} i_1 \wedge i_2 &:= i_1 i_2 \\ i_1 \vee i_2 &:= 1 - (1 - i_1)(1 - i_2) \end{aligned} \tag{1.1}$$

This interpretation has the benefit that i_1 and i_2 can be any real numbers; in particular, they can each be set to the value 1/2.

Evaluating a binary tree with inputs of 1/2 at each of the leaves yields the value of X_n for each particular selection of gates. Considering all possible selections of gates, however, is computationally infeasible for even small trees of small depth. For only the smallest values of n, say $n \leq 5$, can we possible hope to compute X_n for all of the possible choices of gates. Therefore, for medium sized values of n, say $n \leq 20$, we can readily compute the value of X_n for one particular selection of gates for one complete tree of depth n, but we cannot hope to compute X_n for every selection of gates. Therefore, we simply compute X_n on a large number of trees, but we cannot perform an exhaustive investigation of all trees and their associated Boolean functions. For large values of n, say $n \geq 30$, it becomes computationally intractable to even compute the value of X_n for one particular selection of gates on a complete binary tree of depth n. In such cases, we must discriminately choose which gates to evaluate, because we cannot possibly hope to evaluate them all.

In such cases, where we want to approximate the value of X_n on a complete tree of depth n, but where we cannot hope to evaluate all gates of the Boolean tree, we consider a growing tree. We begin simply with the root of a Boolean binary tree. At every stage, we select one leaf of the tree and change it into an internal leaf, by giving it two children and a Boolean gate. Which leaves should be transformed into parent nodes first? We utilize the concept of sensitivity of a leaf to select the next leaf to transform. The leaves that are the most sensitive, i.e., that have the largest potential effect on the evaluation of X_n, should be first.

We rigorously define the notion of the sensitivity of a leaf in a Boolean binary tree. We label the root node as v_0. For a leaf L at depth k in the tree, we write $v_0, v_1, v_2, \ldots, v_k = L$ to describe the path within the tree, from the root node, to the leaf L. For $i \geq 1$, we note that v_{i-1} has two children, namely, v_i and one other child, which we refer to as w_i. Thus v_i and w_i are distinct nodes at level i with the same parent; such nodes are frequently referred to as siblings. We write $X(v)$ to denote the evaluation of the complete Boolean binary subtree that has v as a root. Thus, $X_n = X(v_0)$ is the same X_n that we discussed above. Also, $X(v_1)$ is the evaluation of one of the subtree of the root, and then $X(w_1)$ is the evaluation of the other subtree of the root. So if the root node v_0 of the entire tree is labeled by an AND gate, then $X_n = X_n(v_0) = X(v_1)X(w_1)$. Otherwise, the root node v_0 of the entire tree is labeled by an OR gate, and in this case, $X_n = X_n(v_0) = 1 - (1 - X(v_1))(1 - X(w_1))$. Transversing the tree with the notion in mind, we are naturally lead to the definition of the sensitivity of $L = v_k$. We recall that $v_0, v_1, v_2, \ldots, v_k = L$ is the path through the tree from the root node v_0 to the leaf node $v_k = L$. For ease of notation, we write $g(v_i) = \text{AND}$ or $g(v_i) = \text{OR}$, according to whether node v_i is labeled with an AND or an OR gate. Then the sensitivity of the leaf $v_k = L$ is defined as

$$\begin{aligned} S(v_k) &:= \prod_{i=0}^{k-1} \big([\![g(v_i) = \text{AND}]\!] X(w_i) \\ &\qquad + [\![g(v_i) = \text{OR}]\!] (1 - X(w_i)) \big) \end{aligned} \tag{1.2}$$

where the Iverson notation $[\![A]\!]$ is 1 if event A holds and is 0 otherwise.

We developed a `C++` program to investigate the growth of Boolean binary trees, using the sensitivity of the leaves as a guide for which subtrees to explore first. The program is completely adaptive, according to the sensitivities of the leaves. At each stage of the execution of the program, the most sensitive leaf is chosen, using the definition of sensitivity described above. If several leaves have the same sensitivity, the program selects one of the candidate leaves uniformly at random; sometimes the candidate leaves are at different levels, so this is an important subtlety in the implementation of the program. Once a leaf L is selected to be updated, we consider the path $v_0, v_1, \ldots, v_k = L$ from the root of the tree to the leaf. Only the X-values $X(v_0), X(v_1), \ldots, X(v_k)$ must be updated; this is extremely efficient in terms of the computation required, because at most n nodes are found on the path from the root to the leaf. The sensitivities of every leaf in the tree must be updated afterwards. Recall the definition of $S(v_k)$ in product notation above. Only the X-values $X(w_0), X(w_1), \ldots, X(w_j)$ were changed at this stage, for some value of j, which is usually very small. In other words, only the X-values of the nodes that are ancestors of both the current node and the most sensitive node L must be updated.

We wrote several `C++` programs to perform the computations in this project. Some sample output from the programs is available freely online at `http://www.math.upenn.edu/~ward2/boolean`

We have computed millions and millions of values of X_n for various values of n. For instance, when $n = 15$, we are able to compute approximately 30 values of X_n per second on a 1.42 GHz Power Macintosh G4 computer. We have built a large database that archives all of the output from these investigations. It has grown so large that it is unwieldy to distribute all of it publicly on the Internet, but we summarize some of the results of our computations at the end of this report.

2 Main results

We were inspired to pursue an analysis of X_n because of the Gardy and Woods' intriguing study [7], in which various measures on Boolean functions are analyzed. Gardy and Woods consider trees chosen uniformly among all sub-binary trees with n leaves; they also place randomly assigned logical gates at the internal nodes. We note that a uniformly chosen tree with n leaves is stringy. The typical random function produced in this way is therefore dominated by the $\Theta(1)$ many inputs at leaves of distance $\Theta(1)$ from the root. Their model is natural for some purposes, but we are interested in considering the model in which as $n \to \infty$ the distance from the root to the boundary goes to infinity. For this reason, we consider the simplest such model, namely, the complete binary tree. The typical behavior of a random Boolean function produced by a complete binary tree turns out to be interesting but in some ways elusive.

Besides [7], we note that other recent results about Boolean functions, binary Boolean trees, and tree recurrences, have have been explored in [1], [2], [3], [6], [8], [9], [10], [11], [12],

We recall that X_n is the mean output of a Boolean function defined by a complete Boolean tree of depth n. In this report, we prove the following facts about X_n.

THEOREM 2.1. *The sequence $\{X_n\}$ is a Martingale. With probability 1, we have $\lim_{n\to X_n}$ exists and is either 0 or 1. The moments of X_n may all be computed recursively. In particular, the first four moments of X_n are*

$$\begin{aligned} E(X_n) &= \frac{1}{2} \\ E(X_n^2) &= \frac{1}{2} - \frac{1}{n} + O\left(\frac{\log n}{n^2}\right) \\ E(X_n^3) &= \frac{1}{2} - \frac{3}{2n} + O\left(\frac{\log n}{n^2}\right) \\ E(X_n^4) &= \frac{1}{2} + \frac{\alpha - 2}{n} + O\left(\frac{\log n}{n^2}\right) \end{aligned} \tag{2.3}$$

where $\alpha = \frac{\sqrt{7}-1}{2}$. Since X_n is distributed about $\frac{1}{2}$, it is natural to describe the moments of $Z_n := X_n(1 - X_n)$ as well. We have

$$E(Z_n) = \frac{1}{n + O(\log n)} \tag{2.4}$$

$$E(Z_n^2) \sim \frac{\alpha}{n} \tag{2.5}$$

It follows from this that for some $a > 0$, $P(Z_n \in [a, 1 - a]) = \Theta(1/n)$.

Left open is whether the rest of the time Z_n is typically of order $1/n$ or of some smaller order.

Just as the right $1/n$-tail of Z_n is larger than one might initially expect, it is also not hard to show that the left $1/n$-tail of Z_n is quite small: there is a $c > 0$ such that $P(Z_n < \exp(-cn^2)) > c/n$. We believe in fact that the distribution of $\log Z_n$ is spread over an interval of increasing size as $n \to \infty$. Perhaps, for instance, $\sqrt{\log Z_n}/n$ has a nondegenerate distributional limit.

We point out that there are issues in effective simulation that are bound up with theoretical analyses of the problem. In particular, exact simulation of Z_n (the study of Z_n and X_n is basically interchangeable) requires a time that is exponential in n. However, we have analyzed Z_n extensively (for various n) by approximately simulating Z_n; we do this by exploring only nodes of the tree that one expects to have high impact on the value of Z_n. Given a rooted subtree of already explored nodes (nodes for which we have decided whether the gate is "AND" or "OR"), we define the sensitivity as follows: We write $v_0, v_1, v_2, \ldots, v_k = L$ to denote the path through the tree from the root node v_0 to the leaf node $v_k = L$. For ease of notation, we write $g(v_i) = \text{AND}$ or $g(v_i) = \text{OR}$, according to whether node v_i is labeled with an AND or an OR gate. Then the sensitivity of a leaf is defined as

$$S(v_k) := \prod_{i=0}^{k-1} \big([\![g(v_i) = \text{AND}]\!] X(w_i) + [\![g(v_i) = \text{OR}]\!](1 - X(w_i)) \big) \tag{2.6}$$

where the Iverson notation $[\![A]\!]$ is 1 if event A holds and is 0 otherwise.

At each stage in the growth of the tree, there is a well defined most sensitive remaining node (there may be ties) and one may define a greedy search algorithm which always looks next at the node for which revealing the gate will reduce the variance by the most. It is easy to compute this optimal choice. If one can then compute how close one is to X_n one will know how far to go in order to simulate a pick from X_n with the desired precision. If, further, one can analyze the growth of the exploration tree, then one will know how long it takes to simulate X_n and this will have implications directly on the distribution of X_n. For example, if X_n is typically well approximated by a tree of depth $m < n$, then the distributions of X_n and X_m are close and, if $m = o(n)$, this precludes a limit law with n in the denominator. Obtaining more rigorous results on the growth of the search tree and the accuracy of these approximations is one of our current and ongoing goals.

Ample data generated by various **C++** programs for studying the behavior of X_n when n is small (say, $n \leq 100$) can be obtained from the authors. Our files of data are too large to distribute on the internet at present (we have hundreds of megabytes of files, containing millions of samples of various X_n).

At the present time, it suffices to present a few tables of sample data about X_n at the end of the paper. We give tables of values for X_{15} and X_{20}, using numerical data from millions of samples of X_{15} and X_{20} using **C++** programs that generate sample random Boolean trees. Upon revision of this paper, we plan to present graphical representations of this data, but we hope that the raw data itself is enticing enough for the reviewers at this stage of the project.

3 Analysis and Proofs.

We establish the fact that $\{X_n\}$ is a martingale. We also derive the first four moments of X_n. Using a similar methodology, one can set up similar recurrences and use analogous arguments to derive any of the moments of X_n.

LEMMA 3.1. *The sequence $\{X_n\}$ is a martingale.*

Proof. To see that $\{X_n\}$ is a martingale, suppose that $X_1, \ldots, X_n$ are known. In particular, X_n is known. What is the conditional expectation of X_{n+1} in this case? Recall that, when we compute X_n, we have inputs of $1/2$ for each leaf, all of which occur at depth n. Now we consider the computation of X_{n+1}, given the value of X_n. Each leaf at depth n is replaced with an AND or OR gate, in an IID fashion. The inputs to the leaves at depth $n+1$ are equally likely either 0 or 1, so we input $1/2$ to each leaf at depth $n+1$. Thus, the expected value of each gate output at depth n is also $1/2$. Then the rest of the tree is evaluated exactly as it would be when computing X_n itself, so we obtain

$$E(X_{n+1} \mid X_1, X_2, \ldots, X_n) = X_n$$

and we conclude that the sequence $\{X_n\}$ is a martingale.

COROLLARY 3.1. *With probability 1, $\lim_{n\to\infty} X_n$ exists.*

Proof. Since $0 < X_n < 1$ for each n, we have $E[|X_n|] \leq 1$ for all n. By Lemma 3.1, we know that $\{X_n\}$ is a martingale. Thus, the corollary follows immediately by the Martingale Convergence Theorem (see [4], [5]).

LEMMA 3.2. *We note that*

$$E(X_n) = 1/2$$

is the expected value of X_n.

Proof. The root node is equally likely an AND or OR gate. Writing X_n and $\widetilde{X}_n$ to denote the output of the Boolean functions for the left and right subtrees of the root node, we note that X_n and $\widetilde{X}_n$ are independent and identically distributed. Thus

$$E(X_{n+1}) = \frac{1}{2}E(X_n\widetilde{X}_n) + \frac{1}{2}E(1 - (1 - X_n)(1 - \widetilde{X}_n)) \tag{3.7}$$

Note that X_n takes values on the interval $(0, 1)$, and the distribution of X_n is symmetric about $1/2$. To see this, we observe that if the selection $\vec{g}_n$ of $2^n - 1$ gates results in $X_n = a$, then replacing each AND from $\vec{g}_n$ with an OR, and also replacing the OR's with AND's, we get a new selection of gates that results in $X_n = 1 - a$. Therefore, X_n, $\widetilde{X}_n$, $1 - X_n$, and $1 - \widetilde{X}_n$ all have the same distribution. Thus (3.7) becomes

$$E(X_{n+1}) = \frac{1}{2}E(X_n)^2 + \frac{1}{2} - \frac{1}{2}E(X_n)^2 = \frac{1}{2} \tag{3.8}$$

which completes the proof of the lemma.

We use the following Lemma to aid in the proof of Theorem 3.1. If we define $Z_n = X_n(1 - X_n)$, then we make the following observations.

LEMMA 3.3. *We observe that*

$$E(X_n^2) \nearrow 1/2,$$

i.e., $E(X_n^2)$ increases to the limiting value of $1/2$. To describe the rate of convergence, we note that

$$E(X_n^2) = \frac{1}{2} - \frac{1}{n} + O\left(\frac{\log n}{n^2}\right)$$

Also

$$E(Z_n) := E(X_n(1 - X_n)) = \frac{1}{n} + O\left(\frac{\log n}{n^2}\right)$$

is the expected value of $X_n(1 - X_n)$.

Proof. To see this, we first establish a recurrence for $E(X_n^2)$. When computing X_{n+1}, we again utilize X_n and $\widetilde{X}_n$, namely, the independent random variables which denote the output of the Boolean functions for the left and right subtrees of the root node. If the root contains an AND gate, then $X_{n+1} = X_n\widetilde{X}_n$. On the other hand, if the root contains an OR gate, then $X_{n+1} = 1 - (1 - X_n)(1 - \widetilde{X}_n)$. Thus

$$\begin{aligned} E(X_{n+1}^2) &= \frac{1}{2}E(X_n^2\widetilde{X}_n^2) \\ &\quad + \frac{1}{2}E((1 - (1 - X_n)(1 - X_n^*))^2). \end{aligned} \tag{3.9}$$

As in the previous lemma, we also use the fact that X_n, $\widetilde{X}_n$, $1 - X_n$, and $1 - \widetilde{X}_n$ all have the same distribution. Thus, from (3.9) we obtain

$$\begin{aligned} E(X_{n+1}^2) &= \frac{1}{2}E(X_n^2)^2 + \frac{1}{2} - E(X_n)^2 + \frac{1}{2}E(X_n^2)^2 \\ &= E(X_n^2)^2 + \frac{1}{4} \end{aligned} \tag{3.10}$$

To see that $E(X_n^2)$ is an increasing sequence, note that $(E(X_n^2) - 1/2)^2 \geq 0$, so $E(X_n^2)^2 + 1/4 \geq E(X_n^2)$, or equivalently by (3.10), we have $E(X_{n+1}^2) \geq E(X_n^2)$. Since $E(X_n^2)$ increases and is bounded above by 1, then a limiting value exists; we take a limit on both sides of (3.10) to obtain

$$\lim_{n\to\infty} E(X_{n+1}^2) = (\lim_{n\to\infty} E(X_n^2))^2 + 1/4 \tag{3.11}$$

Thus $\lim_{n\to\infty} E(X_n^2) = 1/2$, which completes the proof of the first statement of the Lemma.

Now we observe

$$\begin{aligned} E(Z_n) &= E(X_n(1 - X_n)) \\ &= E(X_n) - E(X_n^2) \\ &= \frac{1}{2} - E(X_n^2) \end{aligned} \tag{3.12}$$

Thus $E(X_n^2) = \frac{1}{2} - E(Z_n)$. From (3.10), it follows immediately that

$$\frac{1}{2} - E(Z_{n+1}) = \left(\frac{1}{2} - E(Z_n)\right)^2 + 1/4$$

Simplifying, it follows immediately that

$$E(Z_{n+1}) = E(Z_n) - E(Z_n)^2$$

For ease of notation, we write $a_n = E(Z_n)$. So we have $a_{n+1} = a_n - a_n^2$. Then we write $b_n = 1/a_n$, and we compute

$$\begin{aligned} b_{n+1} &= \frac{1}{a_{n+1}} \\ &= \frac{1}{a_n - a_n^2} \\ &= \frac{1}{\frac{1}{b_n} - \frac{1}{b_n^2}} \\ &= \frac{b_n^2}{b_n - 1} \\ &= b_n + 1 + \frac{1}{b_n - 1} \end{aligned} \tag{3.13}$$

Substituting $b_n = b_{n-1} + 1 + \frac{1}{b_{n-1}-1}$ into (3.13) yields

$$b_{n+1} = b_{n-1} + 2 + \sum_{k=n-1}^{n} \frac{1}{b_k - 1}$$

Substituting $b_{n-1} = b_{n-2} + 1 + \frac{1}{b_{n-2}-1}$ yields

$$b_{n+1} = b_{n-2} + 3 + \sum_{k=n-2}^{n} \frac{1}{b_k - 1}$$

and repeated substitutions of a similar flavor eventually yield

$$b_{n+1} = b_1 + n + \sum_{k=1}^{n} \frac{1}{b_k - 1} \tag{3.14}$$

From (3.14), we observe that $b_{n+1} > n$, so $b_k > k - 1$ for all k. Thus, the summation in (3.14) can be bounded by writing

$$\begin{aligned} \sum_{k=1}^{n} \frac{1}{b_k - 1} &= \frac{1}{b_1 - 1} + \frac{1}{b_2 - 1} + \sum_{k=3}^{n} \frac{1}{b_k - 1} \\ &\leq \frac{1}{\frac{16}{3} - 1} + \frac{1}{\frac{256}{39} - 1} + \sum_{k=3}^{n} \frac{1}{(k-1) - 1} \\ &= \frac{3}{13} + \frac{39}{217} + 1 + \sum_{k=2}^{n-2} \frac{1}{k} \\ &= O(\log n) \end{aligned} \tag{3.15}$$

Returning to (3.14), we conclude that

$$b_n = n + O(\log n)$$

Again using (3.14), it follows that

$$b_{n+1} = b_1 + n + \sum_{k=1}^{n} \frac{1}{n + O(\log n)}$$

Therefore, $b_n = n + \log n + O(1)$, and we conclude

$$E(Z_n) = a_n = \frac{1}{n} + O\left(\frac{\log n}{n^2}\right)$$

This proves the final sentence of the lemma. All that remains to show is $E(X_n^2) = 1/2 - 1/n + O\left(\frac{\log n}{n^2}\right)$, but this follows immediately from observing that $E(X_n^2) = E(X_n) - E(X_n(1 - X_n)) = 1/2 - E(Z_n)$.

Now that we have completed the proof of Lemma 3.3, we are equipped to prove the following theorem about the limiting behavior of X_n.

THEOREM 3.1. *With probability 1, $X_n \to 0$ or $X_n \to 1$.*

Proof. It suffices to prove that, for each $\epsilon > 0$, there exists N_ϵ such that

$$\Pr(X_n \in [\epsilon, (1 - \epsilon)]) \le \epsilon$$

for all $n \ge N_\epsilon$.

Note that, if $X_n \in [\epsilon, 1 - \epsilon]$, then also $1 - X_n \in [\epsilon, 1 - \epsilon]$. It follows that $X_n(1 - X_n) \in [\epsilon^2, (1 - \epsilon)^2]$, or equivalently,

$$Z_n \in [\epsilon^2, (1 - \epsilon)^2]$$

So

$$\Pr(X_n \in [\epsilon, (1 - \epsilon)]) \le \Pr(Z_n \in [\epsilon^2, (1 - \epsilon)^2]) \tag{3.16}$$

Note that $E[Z_n]$ is at least $\epsilon^2 \Pr(Z_n \in [\epsilon^2, (1-\epsilon)^2])$, i.e., the expected value of Z_n is at least the probability that Z_n is in the interval $[a, b]$ times the smallest value in the interval, namely a (here, we are using $a = \epsilon^2$ and $b = (1 - \epsilon)^2$). So we obtain

$$\epsilon^2 \Pr(Z_n \in [\epsilon^2, (1 - \epsilon)^2]) \le E[Z_n]$$

Now we return to (3.16) to see that

$$\Pr(X_n \in [\epsilon, (1 - \epsilon)]) \le \frac{1}{\epsilon^2} E[Z_n] \tag{3.17}$$

We just proved in Lemma 3.3 above that $E[Z_n] = \frac{1}{n} + O\left(\frac{\log n}{n^2}\right)$, and it follows that there is some N (depending on ϵ) such that $E[Z_n] < \epsilon^3$ for all $n \ge N_\epsilon$. Therefore

$$\Pr(X_n \in [\epsilon, (1 - \epsilon)]) \le \epsilon \tag{3.18}$$

for all $n \ge N_\epsilon$. This completes the proof of the theorem.

LEMMA 3.4. *We observe that*

$$E(X_n^3) = \frac{1}{2} - \frac{3}{2n} + O\left(\frac{\log n}{n^2}\right)$$

is the third moment of X_n.

Proof. As in Lemma 3.3, we establish a recurrence for $E(X_n^3)$. When computing X_{n+1}, we again write X_n and $\widetilde{X}_n$ to denote the output of the Boolean functions for the left and right subtrees of the root node, which are independent. Then we compute

$$\begin{aligned} E(X_{n+1}^3) &= \frac{1}{2} E(X_n^3 \widetilde{X}_n^3) \\ &\quad + \frac{1}{2} E((1 - (1 - X_n)(1 - \widetilde{X}_n))^3) \end{aligned} \tag{3.19}$$

We once again use the fact that X_n, $\widetilde{X}_n$, $1 - X_n$, and $1 - \widetilde{X}_n$ share a common distribution. Thus

$$\begin{aligned} E(X_{n+1}^3) &= \frac{1}{2} E(X_n^3)^2 + \frac{1}{2} - \frac{3}{2} E(X_n)^2 \\ &\quad + \frac{3}{2} E(X_n^2)^2 - \frac{1}{2} E(X_n^3)^2 \\ &= \frac{3}{2} E(X_n^2)^2 + \frac{1}{8} \end{aligned} \tag{3.20}$$

recall from (3.10) that

$$E(X_{n+1}^2) = E(X_n^2)^2 + \frac{1}{4} \tag{3.21}$$

Plugging this result into (3.20) yields

$$\begin{aligned} E(X_{n+1}^3) &= \frac{3}{2}\left(E(X_{n+1}^2) - \frac{1}{4}\right) + \frac{1}{8} \\ &= \frac{3}{2} E(X_{n+1}^2) - \frac{1}{4} \end{aligned} \tag{3.22}$$

and by Lemma 3.3, we conclude that

$$E(X_n^3) = \frac{1}{8} - \frac{3}{4n} + O\left(\frac{\log n}{n^2}\right) \tag{3.23}$$

This establishes the lemma.

We abbreviate the proof of the next theorem [and may expand the proof for the final submission before publication]. We recall that

$$Z_n = X_n(1 - X_n) \tag{3.24}$$

Using the lemmas above, we now establish the following asymptotics for $E(Z_n^2)$.

THEOREM 3.2. *The second moment of Z_n^2 decays as $E(Z_n^2) \sim \frac{\alpha}{n}$ where $\alpha = \frac{\sqrt{7}-1}{2} \approx .82$.*

Proof. As in several of the above lemmas, we observe that

$$E(X_{n+1}^4) = \frac{1}{2}E(X_n^4\widetilde{X}_n^4) + \frac{1}{2}E((1-(1-X_n)(1-\widetilde{X}_n))^4) \tag{3.25}$$

where, as above, X_n and $\widetilde{X}_n$ denote the output of the Boolean functions for the left and right subtrees of the root node. Simplifying, and using the same methodology as in the lemmas (i.e., utilizing the independence of X_n and $\widetilde{X}_n$, and also the fact that X_n, $1-X_n$, $\widetilde{X}_n$, and $1-\widetilde{X}_n$ have the same distribution), and using the results established in Lemmas 3.3 and 3.4, it follows that

$$E(X_{n+1}^4) = E(X_n^4)^2 - \frac{3}{2}E(X_n^2)^2 + \frac{3}{2}E(X_n^2) - \frac{1}{8} \tag{3.26}$$

In order to simplify things, we define

$$h_n := \frac{1}{2} - E(X_n^4) \tag{3.27}$$

and

$$d_n := \frac{1}{2} - E(X_n^2) \tag{3.28}$$

Then it follows from (3.26) that

$$h_n = h_{n-1} - h_{n-1}^2 + \frac{3}{2}d_n^2 \tag{3.29}$$

We recall that, by Lemma 3.3, $d_n = \frac{1}{n} + O\left(\frac{\log n}{n^2}\right)$, and it follows that $h_n \sim \frac{\alpha}{n}$ for some α. To solve for α, we rewrite (3.29) as

$$\begin{aligned} \frac{\alpha}{n} &\sim \frac{\alpha}{n-1} - \frac{\alpha^2}{(n-1)^2} + \frac{3}{2}\frac{1}{(n-1)^2} \\ &= \frac{\alpha}{n-1} - \frac{\alpha}{(n-1)^2} + \frac{\alpha - \alpha^2 + 3/2}{(n-1)^2} \end{aligned} \tag{3.30}$$

so we require $-\alpha^2 + \alpha + 3/2 = 0$, and we conclude that $\alpha = \frac{\sqrt{7}-1}{2} \approx .82$. This verifies the theorem.

We observe that

Corollary 3.2. *It follows from the theorem above that the fourth moment of X_n*

$$E(X_n^4) = \frac{1}{2} + \frac{\alpha - 2}{n} + O\left(\frac{\log n}{n^2}\right)$$

where $\alpha = \frac{\sqrt{7}-1}{2}$.

Proof. We note that $E(X_n^4) = E(Z_n^2) - E(X_n^2) + 2E(X_n^3)$, and then the corollary follows immediately from Lemmas 3.3 and 3.4 along with Theorem 3.2.

4 Experimental Data

If we write $i = P(X_{15} \leq a)$, then the following chart gives the values of i and analogous a value. So, for example, we see that 10% of the time, we have $X_{15} \leq 4.23 \times 10^{-9}$. The data is based upon four million samples of X_{15}. We note that X_{15} cannot actually take on the values 0 or 1 (in other words, $0 < X_{15} < 1$ always), but some values of X_{15} that were simulated by **C++** became so close to 0 or 1 that the program did not have sufficient accuracy to make a distinction.

We also give similar data for X_{20}, based on 800,000 samples.

We emphasize that each sample of X_{15} and X_{20} was produced by computing a complete Boolean binary tree of depth 15 and 20, respectively.

On the other hand, complete Boolean binary trees of larger depth, say depth 100, are impossible to sample completely. So we have an interactive **C++** program that allows the user to sample values from X_{100}, for instance, with interactions about when to stop the simulation. The **C++** program is trained to stop the simulation itself if it detects that the sensitivities of the leaf nodes, collectively, are sufficiently small.

The **C++** program has several other features. For instance, it lets us visualize the data by examining the profile as the tree grows. The evolution of the profile as the most sensitive nodes are selected within the tree is a fascinating phenomenon. Besides further studying the profile, we also plan to continue investigating stopping criteria for the growth of large Boolean binary tree when simulating X_n for large n, for example, $n = 100$.

Values of $i = P(X_{15} \leq a)$ for various i's, based on four million samples of X_{15}:

i	a
0	0
0.02	$1.5842880459137615782 \times 10^{-18}$
0.04	$1.5417759331674780113 \times 10^{-14}$
0.06	$3.6853441307647837367 \times 10^{-12}$
0.08	$1.946150783972689839 \times 10^{-10}$
0.1	$4.2389795053727498602 \times 10^{-09}$
0.12	$5.5574572680172713111 \times 10^{-08}$
0.14	$4.8096310496682111124 \times 10^{-07}$
0.16	$3.0620598368277611197 \times 10^{-06}$
0.18	$1.5096759165431513337 \times 10^{-05}$
0.2	$6.0584853376422378275 \times 10^{-05}$
0.22	0.00020423085893656053673
0.24	0.00060358903651591916482
0.26	0.0015788697227436521413
0.28	0.0037028015298202639725
0.3	0.0081775785999889938349
0.32	0.016096791381125210435
0.34	0.02931675216108991372
0.36	0.049929145494566826158
0.38	0.080863856150230561948
0.4	0.12284620845589767912
0.42	0.17824045429117807426
0.44	0.24535675616933050325
0.46	0.32363312838019353546
0.48	0.40902018329138667418
0.5	0.49958615335734124496
0.52	0.59031949427051666479
0.54	0.67577470773851533448
0.56	0.75413287054239663831
0.58	0.82141239443075020343
0.6	0.87690879166932234057
0.62	0.91887054992423133903
0.64	0.94994420606753837699
0.66	0.9705969104376125367
0.68	0.9838663676699012095
0.7	0.9917975019371320089
0.72	0.99628960071692485023
0.74	0.99841700621740403498
0.76	0.99939369149401702241
0.78	0.99979514056467388983
0.8	0.99993933648367427924
0.82	0.99998486455164625752
0.84	0.9999969351429923714
0.86	0.99999951878718318365
0.88	0.99999994442444495313
0.9	0.99999999576808529245
0.92	0.9999999998065358664
0.94	0.99999999999631150605
0.96	0.99999999999998467892
0.98	1

Values of $i = P(X_{20} \leq a)$ for various i's, based on 800,000 samples of X_{20}:

i	a
0	0
0.02	$5.0382163087912427022 \times 10^{-38}$
0.04	$1.5230128229556538674 \times 10^{-27}$
0.06	$6.3286657458441326702 \times 10^{-22}$
0.08	$5.1735411059772937349 \times 10^{-18}$
0.1	$2.9976021664879226591 \times 10^{-15}$
0.12	$3.9904445455628754719 \times 10^{-13}$
0.14	$2.4095577834622933117 \times 10^{-11}$
0.16	$7.7836396180672318864 \times 10^{-10}$
0.18	$1.4707447536456487624 \times 10^{-08}$
0.2	$1.8935175488311919258 \times 10^{-07}$
0.22	$1.7570854901545337144 \times 10^{-06}$
0.24	$1.1945313073671176302 \times 10^{-05}$
0.26	$6.2980668432316506337 \times 10^{-05}$
0.28	0.00027352357944943062051
0.3	0.00097795102835056854466
0.32	0.0029612089558833220443
0.34	0.0078629214644783118615
0.36	0.018220237574111095707
0.38	0.037505430862690404548
0.4	0.070754484857089683381
0.42	0.12213471928528014943
0.44	0.1920888875815146557
0.46	0.28193889046471437565
0.48	0.38713351770430382004
0.5	0.49943452288831491348
0.52	0.61368509464475073933
0.54	0.71905791401265817253
0.56	0.80821918187350394458
0.58	0.87741511277304573557
0.6	0.92859840337232846252
0.62	0.96225347983792974826
0.64	0.9815937200809048413
0.66	0.99203387517333319057
0.68	0.99695351480120708576
0.7	0.99898484968621981128
0.72	0.99971894535199090637
0.74	0.99993478528828672047
0.76	0.99998779094032930193
0.78	0.99999819053417760006
0.8	0.99999979618479006849
0.82	0.99999998367149167677
0.84	0.99999999910327408426
0.86	0.99999999997058586221
0.88	0.99999999999949451546
0.9	0.9999999999999960032
0.92	1
0.94	1
0.96	1
0.98	1

Acknowledgments

We appreciate the input of Svante Janson, who simultaneously derived several of the observations presented here. We also acknowledge Bob Sedgewick's insightful advice about using randomization in the implementation of data structures (*Finding Paths in Graphs*, A of A 2005).

References

[1] C. Banderier, M. Bousquet-Mélou, P. Flajolet A. Denise, D. Gardy, and D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Mathematics*, 246:29–55, 2002.

[2] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002.

[3] B. Chauvin, P. Flajolet, D. Gardy, and B. Gittenberger. And/or tree revisited. *Combinatorics, Probability and Computing*, 13(4–5):475–497, 2004.

[4] R. Durrett. *Probability: Theory and Examples.* Duxbury, Belmont, CA, 3rd edition, 2005.

[5] W. Feller. *An Introduction to Probability Theory and Its Applications.* Wiley, New York, 1968, 1971.

[6] J. Fill, P. Flajolet, and N. Kapur. Singularity analysis, hadamard products, and tree recurrences. *Journal of Computational and Applied Mathematics*, 174:271–313, February 2005.

[7] D. Gardy and A. Woods. And/or tree probabilities of boolean functions. In Conrado Martínez, editor, *2005 International Conference on Analysis of Algorithms*, volume AD of *DMTCS Proceedings*, pages 139–146. Discrete Mathematics and Theoretical Computer Science, 2005.

[8] H. Lefmann and P. Savický. Some typical properties of large and/or Boolean formulas. *Random Structures and Algorithms*, 10:337–351, 1997.

[9] P. Savický. Bent functions and random Boolean formulas. *Discrete Mathematics*, 147:211–237, 1995.

[10] P. Savický. Complexity and probability of some Boolean formulas. *Combinatorics, Probability and Computing*, 7(4):451–463, 1998.

[11] P. Savický and A. Woods. The number of Boolean functions computed by formulas of a given size. *Random Structures and Algorithms*, 13(3–4):349–382, 1998.

[12] I. Wegener. *The complexity of Boolean functions.* Teubner, Stuttgart, 1987.