Probabilistic Boolean Logic

Lakshmi N. B. Chakrapani, Krishna V. Palem

College of Computing

Georgia Institute of Technology

Atlanta, Georgia, USA.

lnc@gatech.edu

In this paper, we introduce and define *Probabilistic Boolean Logic*, whose logical operators are "correct" with a probability $0 < p \leq 1$. Analogous to conventional Boolean logic, we define well-formed probabilistic Boolean formulae (PBF). Every PBF is associated with two attributes, the underlying Boolean function it computes, and a for a specific input, the probability that this boolean function is computed correctly. To characterize these attributes simultaneously, we introduce the concept of a *sample space generator* associated with any PBF. For a specific input, a sample space generator of a PBF *generates* a sample space, which defines the random experiment which determines the value of the PBF. Using the notion of sample space generators, we define equivalence of two PBF, derive identities and properties. Based on properties of sample space generators, we prove that for any probabilistic boolean function, there is a probabilistic boolean formula which computes it and vice versa. We introduce and relate probabilistic boolean circuits to classical models of computation, such as randomized circuits and probabilistic automata. Synthesis and optimization of probabilistic boolean circuits from specifications (in the form of probabilistic boolean functions) has implications to circuit design with unreliable logic gates. It has been experimentally demonstrated earlier that such circuits which utilize unreliable logic gates, can dramatically reduce energy consumption of certain applications and are vital towards sustaining Moore's law into the next decades. Extending this, we derive a theoretical result to prove that in the domain of CMOS, transition functions of probabilistic automata maybe realized with lesser energy complexity by probabilistic circuits when compared to randomized circuits of identical size and depth.

This work is supported in part by DARPA under seedling contract #F30602-02-2-0124, by the DARPA ACIP program under contract #FA8650-04-C-7126 through a subcontract from USC-ISI and by an award from Intel Corporation.

Categories and Subject Descriptors: []:

General Terms:

Additional Key Words and Phrases:

1. INTRODUCTION, RELATED WORK AND ROADMAP

The very notion of automated computation, the machines that can perform such computation and the languages to program such machines has its roots in the study and advances in logic (see [Davis 2001] for an excellent overview and a historical perspective, which relates advances in logic to the birth of "modern" computers and computer science in their present form). Narrowly and considering one aspect of computers, Boolean logic and the circuit model of computation based on boolean logic have spurred advances in the specification, and automated construction of silicon-based digital VLSI circuits (which conventional computers are built from). Concurrently, in the domain of computer-science, specifically in the domain of algorithms, probability has played a vital role from Rabin's work on probabilistic automata and probabilistic algorithms [Rabin 1963; 1976] and Chaitin and Schwartz's work on information theory [Chaitin and Schwartz 1978], to name a few. These seminal contributions have opened up vast areas of study, by incorporating probability and randomness into consideration. Models of computation which incorporate probability into consideration (on which these algorithms execute), are typically based on deterministic logical operations ("steps" or "gates" for example), which consume "coin tosses" or random bits. We shall refer to such models of computation as "explicitly" probabilistic. Models of computation which are "implicitly" probabilistic—where elements such as gates were susceptible to erroneous behavior—were chiefly studied in the context of unreliable computing elements with an aim of removing (or reducing) such probabilistic (unreliable) behavior. For example, von-Neumann's seminal work [von Neumann 1956] was inspired by the need for implementing reliable computing in the presence of faults. More recently, Pippenger shows how boolean functions may be computed reliably (with constant multiplicative redundancy) by gates susceptible to noise [Pippenger et al. 1991; Pippenger 1985; 1989] and in the domain of CMOS, Bahar et al. demonstrate methods for improving the noise immunity of logic circuits by adopting design styles based on Markov Random Fields [Bahar et al. 2003; Nepal et al. 2005].

The ability of conventional Boolean logic to specify and model the behavior of digital circuits is increasingly challenged (due to reasons elaborated below). Motivated by this and a desire to reason about probability and Boolean logic in an unified model, in this work, we study *Probabilistic Boolean Logic* and an implicitly probabilistic model of computation based on this logic. In this model, we capture Boolean logic as well as probability in a unified model, through a probabilistic extension to boolean logic where the three canonical operators—conjunction, disjunction and negation—have an associated probability p $(0 < p \leq 1)$ of "correctness". Formulae are composed of probabilistic boolean operators, boolean variables and the constants $0, 1$. For any valid input to a probabilistic boolean formula (PBF), the value of a PBF is the outcome of a random experiment, whose sample space is determined by (i) the input and the operators in the PBF (ii) their associated probabilities of correctness. To simultaneously capture these attributes, and to "interpret" probabilistic boolean formulae, we introduce the concept of a *sample space generator*, which is a set of events, each associated with a (deterministic) boolean formula. For a specific input, the sample space generator can be used to construct the sample space which determines the value of the PBF. Subsequently, we introduce and study *Probabilistic Boolean Circuits*, a model of computation based on probabilistic boolean logic.

The study of probabilistic boolean logic and circuits and relating them to classical models of computation—including the celebrated probabilistic automata model—is the first contribution of this work. One may be led to thinking that this logic and consequently, the computational model based on this logic, are equivalent to a conventional model of computation with input coin tosses (models such as the randomized circuit model [Motwani and Raghavan 1995] are good examples). However, we show that probabilistic boolean circuits, composed of "implicitly" probabilistic gates, may be identical to randomized circuits (employing "explicit" randomness) in terms of conventional complexity measures like size and depth, but differ in energy consumption in any physical implementation, because of thermodynamic reasons. This is the second contribution of this work. This result is an extension of prior work: (i) A theoretical result which showed such a separation in the energy complexity between probabilistic algorithms and deterministic algorithms (of identical time complexity) in the BRAM model of computation [Palem 2003; 2005] and (ii) An empirical demonstration that CMOS circuits which are susceptible to noise—and hence "unreliable"—can implement certain class of applications in an energy efficient manner when compared to "reliable" circuits which realize equivalent functionality [Chakrapani et al. 2006; George et al. 2006; Chakrapani et al. 2007]. Such study of implicitly probabilistic models of computation and their practical implementation gain renewed urgency, based on recent trends dictated by Moore's law which indicate a shift from "deterministic" and "reliable" CMOS devices to those whose behavior is "statistical" or "unreliable". For example, the ITRS road map forecasts [ITRS 2002] *"Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects."* This probabilistic behavior of CMOS building blocks in future technology generations, and its likely impact on computing platforms has been recognized by researchers in the industry;

to cite an example, Borkar notes *We will shift from the deterministic designs of today to probabilistic and statistical designs of the future$\cdots$ So we now say, "If I do this in the design, the transistors and therefore the chip will perform in this way." In the future, we will say, "If I design with this logic depth or this transistor size, I will increase the probability that a given chip will perform in this way."*. This has led to a renewed interest in incorporating probability into CMOS device and architecture models. Since probabilistic boolean circuits can serve as a model for VLSI logic circuits which are susceptible to transient noise (and faults), it can be used as a sound theoretical basis for advances in the construction of "statistical VLSI designs". The foundations introduced in this paper enable us to characterize, synthesize and optimize probabilistic boolean circuits from probabilistic specifications. This can have a dramatic impact on energy consumption of VLSI circuits [Chakrapani et al. 2007; George et al. 2006] and can help sustain Moore's law into the next decade [Moore's Law ; Korkmaz et al. 2006]. This is the third contribution of this work.

1.1 Roadmap

2. PROBABILISTIC BOOLEAN FORMULAE

In this section, we introduce probabilistic boolean formulae and their properties. We first need to define the notion of *equivalence* of two (deterministic) boolean formulae. Informally, two boolean formulae are considered to be equivalent, if for any assignment of boolean constants to the variables in the formulae, the value of the formulae are identical. For any boolean formula F consider the linearly ordered set of variables in F, denoted by VAR_F and let $|\text{VAR}_F| = k$. Let the set of all $k-$ bit binary strings be denoted by $\{0,1\}^k$. Consider any $I \in \{0,1\}^k$. The value of the formula F when the j^{th} bit of I is assigned to the j^{th} variable in VAR_F will be referred to as "the value of F for an input I" or denoted by $F(I)$.

Informally, probabilistic boolean formulae—like their deterministic counterparts—can be constructed from the boolean constants $0, 1$, boolean variables, and the three canonical boolean operators: conjunction, disjunction and negation. In addition, each of these boolean operators are associated with a parameter p where $0 < p \leq 1$ which defines its "probability of correctness". Initially, we consider only rational probabilities of correctness, though relaxing this requirement introduces several interesting properties. Henceforth in the sequel, unless specified otherwise, any probability value p will be defined to be $0 < p \leq 1$ and $p \in \mathbb{Q}$.

Analogous to well formed boolean formulae, valid probabilistic boolean formulae can be defined as follows:

(1) Any boolean variable and the values 0,1 are valid probabilistic boolean formulae. Typically we shall denote boolean variables using lower case alphabets.

(2) If F, G are valid probabilistic boolean formulae, $(F \vee_p G)$, $(\neg_p F)$ and $(F \wedge_p G)$ are valid probabilistic boolean formulae.

Henceforth, we will use the term "probabilistic boolean formula", or the notation PBF to mean "valid probabilistic boolean formula". In addition, the length of a probabilistic boolean formula would refer to the total number of operators in the formula.

2.1 Sample Space

The previous section introduced the concept of a PBF. Intuitively, for any assignment of values to the variables in a PBF, the value of the PBF is determined by (i) the boolean operators in the PBF (ii) the probabilities of correctness of each of the boolean operators in the PBF. Whereas the former defines the "underlying" (deterministic) boolean function, the latter characterizes the random experiment which determines if this boolean function is evaluated "correctly". Note that the probability that the underlying boolean function is evaluated correctly might vary with the input (and indeed it does). We will first consider the random experiment associated with a probabilistic boolean formula and a fixed input and characterize its sample space.

Consider a PBF $F = (x \vee_p y)$, where x, y are boolean variables. Consider an input $I \in \{0, 1\}^2$. In particular let $I =< 1, 0 >$ where 1 is assigned to x and 0 to y. There exists a sample space, denoted by $\mathcal{S}_{F(I)}$, where $(p)|\mathcal{S}_{F(I)}|$ sample points in $\mathcal{S}_{F(I)}$ correspond to the event $F \equiv 1$, which corresponds to the event $\vee_p$ is evaluated "correctly" and hence the value of F is $(1 \vee 0) \equiv 1$. Further, $(1 - p)|\mathcal{S}_{F(I)}|$ sample points in $\mathcal{S}_{F(I)}$ correspond to the event $F \equiv 0$, which corresponds to the event $\vee_p$ is evaluated "incorrectly" and hence the value of F is $\neg(1 \vee 0) \equiv 0$. For convenience we refer to the former as "sample points of the *type* 1" and the latter as "sample points of the *type* 0". We observe that the value of F is associated with a sample space for each of the four inputs $< 0, 0 >$, $< 0, 1 >$, $< 1, 0 >$ and $< 1, 1 >$, furthermore, the sample points in each of these sample spaces are of type 0 or are of type 1.

The value of a PBF $F = (x \vee_p y)$ for an input I, denoted by $F(I)$, is equal to the outcome of a random experiment whose sample space is $\mathcal{S}_{F(I)}$. This captures the notion that "for an input I, there is a probability p that the value of the probabilistic boolean formula $(x \vee_p y)$ is equal to the value of $(1 \vee 0)$ and a probability $(1 - p)$ that it is equal to the value of $\neg(1 \vee 0)$". Or, in simpler terms, there is a probability p that the value of $x \vee_p y$ is "correct" and a probability $1 - p$ that it is "incorrect" (where the "correct" value is defined to be the value of $x \vee y$). Correspondingly, sample spaces of F when F is of the form $(x \wedge_p y)$ and $(\neg_p x)$ can be defined in a similar fashion. Consider a PBF of length $k + 1$ where $k > 0$ and let it be denoted by H. Since H is a valid probabilistic boolean formula, H is of the form $(F \vee_p G)$ or $(F \wedge_p G)$ or $(\neg_p F)$ where F and G are of length k or less. Let $H \equiv (F \vee_p G)$ where $p = \frac{m}{n}$ and let $\mathcal{S}_{F(I)}$ and $\mathcal{S}_{G(I)}$ be the sample spaces of F and G for any input I to H. The sample space of H for an input I denoted by $\mathcal{S}_{H(I)}$, can be constructed as follows. Initially let $\mathcal{S}_{H(I)} \equiv \phi$. For any $(s_i, \hat{s}_j) \in \mathcal{S}_{F(I)} \times \mathcal{S}_{G(I)}$, where s_i is of type t_i and $\hat{s}_j$ is of type $\hat{t}_j$, add m sample points of the type $(t_i \vee \hat{t}_j)$ to $\mathcal{S}_{H(I)}$ and $n - m$ sample points of the type $\neg(t_i \vee \hat{t}_j)$ to $\mathcal{S}_{H(I)}$. This captures the notion that given F evaluates to t_i and G evaluates to t_j, the value of H is "correct" with a probability $\frac{m}{n}$ and "incorrect" with the probability $1 - \frac{m}{n}$. The technique for constructing sample space for the case where $H \equiv F \wedge_p G$ and $H \equiv \neg_p F$ is similar. Since sample spaces exist for valid inputs to PBF of size 0 and 1, sample spaces exist for valid inputs to any valid PBF of length $k > 1$.

We note that the given a PBF and an input to the PBF, the corresponding sample space is not unique, since several sample spaces may be *equivalent*. Consider two sample spaces $\mathcal{S}$ and $\mathcal{S}'$. By definition, sample points in these sample spaces are of type 0 or 1. Let $\mathcal{S}_{\text{type}=0} \subseteq \mathcal{S}$ be the set of sample points in $\mathcal{S}$ of type 0 and correspondingly let $\mathcal{S}'_{\text{type}=0} \subseteq \mathcal{S}'$ be the set of sample points in $\mathcal{S}'$ of type 0. The $\mathcal{S}$ is equivalent to $\mathcal{S}'$, denoted by $\mathcal{S} \equiv \mathcal{S}'$ if

$$\frac{|\mathcal{S}_{\text{type}=0}|}{|\mathcal{S}|} = \frac{|\mathcal{S}'_{\text{type}=0}|}{|\mathcal{S}'|} \tag{1}$$

2.2 Sample Space Generator

As noted before, for any probabilistic boolean formula $F = (x \vee_p y)$, a sample space exists for each of the valid inputs(in this case, four) to F. We may now define the notion of a *sample space generator* $\mathcal{S}_F$ which will serve as a model to provide semantics to PBF.

Definition 2.1. A sample space generator $\mathcal{S}_F$ of a probabilistic boolean formula F, is a set of sample points such that any sample point $s_i \in \mathcal{S}_F$ is of type t_i, where t_i is a boolean formula. Furthermore,

(1) For any $s_i \in \mathcal{S}_F$ which is of type t_i the set of boolean variables in t_i is a subset of $\text{VAR}_F \cup \{0, 1\}$, where VAR_F is the set of boolean variables in F.

(2) For any valid input I to F, the set $\{$*sample point of the type* $t_i(I) : s_i \in \mathcal{S}_F\} \equiv \mathcal{S}_{F(I)}$. Here $t_i(I)$ is used to denote the value of the boolean formula t_i, where for an assignment from $\{0, 1\}$ to a boolean variable $x \in \text{VAR}_F$, an identical assignment is made to x if $x \in \text{VAR}_{t_i}$

In other words, the sample space generator $\mathcal{S}_F$ associated with a probabilistic boolean formula F, can be used to *generate* the sample space $\mathcal{S}_{F(I)}$ for any input I to F, by evaluating the individual boolean formula associated with each of the sample points in $\mathcal{S}_F$ to yield $\mathcal{S}_{F(I)}$. In general, given a probabilistic boolean formula, many distinct sample space generators might satisfy the conditions outlined above. We outline the rules for computing one such sample space generator (which we will refer to as the "canonical" sample space generator). Let H be a probabilistic boolean formula of the form $H = (F \vee_p G)$ where F, G

are probabilistic boolean formulae and $\mathcal{S}_F$, $\mathcal{S}_G$ are the sample space generators which correspond to F, G respectively. Trivially, the sample space generator of a boolean formula of the type $F \equiv x$ where x is a boolean variable, is a set with one sample point of the type x. The sample space generator $\mathcal{S}_H$ which corresponds to the probabilistic boolean formula H can be constructed as follows

Let $p = \frac{m}{n}$ where m, n are relatively prime. Let

$$\begin{aligned}
\hat{\mathcal{S}}_{Hc} &\equiv \{\text{sample points of type } (t_i \vee t_j) : s_i \in \mathcal{S}_F, s_j \in \mathcal{S}_G\} \\
\mathcal{S}_{Hc} &= \{m \text{ sample points, each of the type } t_k : s_k \in \hat{\mathcal{S}}_{Hc}\} \\
\hat{\mathcal{S}}_{Hw} &\equiv \{\text{sample points of the type } \neg(t_i \vee t_j) : s_i \in \mathcal{S}_F, s_j \in \mathcal{S}_G\} \\
\mathcal{S}_{Hw} &= \{(n-m) \text{ sample points, each of the type } t_l : s_l \in \hat{\mathcal{S}}_{Hw}\} \\
\mathcal{S}_H &= \mathcal{S}_{Hc} \cup \mathcal{S}_{Hw}
\end{aligned}$$

To illustrate, consider a probabilistic boolean formula $F = (x \vee_p y)$. Let $p = \frac{m}{n}$ such that m and n are relatively prime, sample space generator $\mathcal{S}_F$ of F is constructed as follows. Let $\mathcal{S}_{Fc}$ be a set such that $|\mathcal{S}_{Fc}| = m$ and for any sample point $s_k \in \mathcal{S}_{Fc}$, s_k is of the type $(x \vee y)$. Similarly, let $\mathcal{S}_{Fw}$ be a set such that $|\mathcal{S}_{Fw}| = (n-m)$ and for any sample point $s_l \in \mathcal{S}_{Fw}$, s_l is of the type $\neg(x \vee y)$. Then,

$$\mathcal{S}_F = \mathcal{S}_{Fc} \cup \mathcal{S}_{Fw} \tag{2}$$

We have illustrated the technique to construct sample space generators of probabilistic boolean formulae of the type $H = (F \vee_p G)$ where F and G are probabilistic boolean formulae. This can trivially be extended to probabilistic boolean formulae of the type $H = (F \wedge_p G)$ as follows: Let $p = \frac{m}{n}$ where m, n are relatively prime.

$$\begin{aligned}
\hat{\mathcal{S}}_{Hc} &\equiv \{\text{sample points of type } (t_i \wedge t_j) : s_i \in \mathcal{S}_F, s_j \in \mathcal{S}_G\} \\
\mathcal{S}_{Hc} &= \{m \text{ sample points, each of the type } t_k : s_k \in \hat{\mathcal{S}}_{Hc}\} \\
\hat{\mathcal{S}}_{Hw} &\equiv \{\text{sample points of the type } \neg(t_i \wedge t_j) : s_i \in \mathcal{S}_F, s_j \in \mathcal{S}_G\} \\
\mathcal{S}_{Hw} &= \{(n-m) \text{ sample points, each of the type } t_l : s_l \in \hat{\mathcal{S}}_{Hw}\} \\
\mathcal{S}_H &= \mathcal{S}_{Hc} \cup \mathcal{S}_{Hw}
\end{aligned}$$

Similarly for probabilistic boolean formulae of the type $H = (\neg_p F)$ where F is a probabilistic boolean formula, the sample space generator can be constructed as follows. Let $p = \frac{m}{n}$ where m, n are relatively prime.

$$\begin{aligned}
\hat{\mathcal{S}}_{Hc} &\equiv \{\text{sample points of type } (\neg t_i) : s_i \in \mathcal{S}_F\} \\
\mathcal{S}_{Hc} &= \{m \text{ sample points, each of the type } t_k : s_k \in \hat{\mathcal{S}}_{Hc}\} \\
\hat{\mathcal{S}}_{Hw} &\equiv \{\text{sample points of the type } t_i : s_i \in \mathcal{S}_F\} \\
\mathcal{S}_{Hw} &= \{(n-m) \text{ sample points, each of the type } t_l : s_l \in \hat{\mathcal{S}}_{Hw}\} \\
\mathcal{S}_H &= \mathcal{S}_{Hc} \cup \mathcal{S}_{Hw}
\end{aligned}$$

Consider F, a PBF of length 1. Hence, F is of the form $\neg_p x$, $(x \vee_p y)$ or $(x \wedge_p y)$ where x and y are boolean variables or constants. By exhaustive listing of cases it can be demonstrated that

CLAIM 2.2.1. *For any valid input I to a* PBF *F of length 1, if $\mathcal{S}_F$ is the canonical sample space generator constructed using the method above, the set* $\{\text{sample point of the type } t_i(I) : s_i \in \mathcal{S}_F\} \equiv \mathcal{S}_{F(I)}$

THEOREM 2.2. *For every probabilistic boolean formula F, there exists a valid sample space generator $\mathcal{S}_F$ associated with F*

PROOF. By induction
Valid sample space generators exist for all boolean variables and the values 0, 1 and all boolean formulae of length one (claim 2.2.1). Let H be of length $k+1$ where $k \geq 1$. Assume that valid sample space generators exist for all boolean formulae of length k or less. Then, from the definition of valid probabilistic boolean formula, H is of the form $(\neg_p F)$, $(F \vee_p G)$ or $(F \wedge_p G)$ where G and F are probabilistic boolean formulae of length k or less.

Assume that H is of the form $(F \vee_p G)$ (the proof for the other cases are similar). Let $p = \frac{m}{n}$ where m and n are relatively prime. Let the sample space generator of H be constructed using the method described in Section 2.1 and let it be $\mathcal{S}_H$. By assumption, the sample space generators of F and G exist and are $\mathcal{S}_F$ and $\mathcal{S}_G$ respectively. For any valid input I to H, the sample spaces $\mathcal{S}_{F(I)}$ and $\mathcal{S}_{G(I)}$, generated by $\mathcal{S}_F$ and $\mathcal{S}_G$ respectively, characterize F and G respectively, since $\mathcal{S}_F$ and $\mathcal{S}_G$ are valid sample space generators (from inductive assumption). $\mathcal{S}_{F(I)}$ and $\mathcal{S}_{G(I)}$ are composed of events of two types, 0 and 1 (from definition of sample spaces).

Consider $\mathcal{S}_{F(I)} \times \mathcal{S}_{G(I)}$. For any $(s_i, s_j) \in \mathcal{S}_{F(I)} \times \mathcal{S}_{G(I)}$ where s_i, s_j are of type $t_i, t_j \in \{0, 1\}$ respectively, m sample points in $\mathcal{S}_{H(I)}$ are of type $t_i \vee t_j$ (from construction). Correspondingly $(n-m)$ sample points in $\mathcal{S}_{H(I)}$ are not of type $t_i \vee t_j$. Thereby, for any input I to H, the fraction of the number of sample points in $\mathcal{S}_{H(I)}$ equal to the value of $F(I) \vee G(I)$ is $\frac{m}{n} = p$. This is equivalent to the sample space of $H(I)$ and hence $\mathcal{S}_{H(I)}$ is a sample space which characterizes the value of H. Hence $\mathcal{S}_H$ is a valid sample space generator. □

3. PROPERTIES OF PROBABILISTIC BOOLEAN FORMULAE

In this section, we explore a few properties of probabilistic boolean formulae by studying the properties of the underlying sample space generators. In particular, we show that probabilistic boolean formulae are commutative but not associative.

3.1 Commutativity

Consider a PBF H of the form $(x \vee_p y)$, where x, y are boolean variables. Let $\mathcal{S}_H$ be the sample space generator of H. The sample points in $\mathcal{S}_H$ are either of type $x \vee y$ or $\neg(x \vee y)$, furthermore $x \vee y \equiv y \vee x$ and $\neg(x \vee y) \equiv \neg(y \vee x)$, it immediately follows that

THEOREM 3.1. *The operator $\vee_p$ is commutative*

Corollary 3.2. The operator $\wedge_p$ is commutative.

3.2 Associativity

Probabilistic boolean formulae, in general are not associative. This can be demonstrated by assuming that probabilistic boolean formulae are associative and constructing a simple counter-example. However, we shall construct a more complex proof with an aim of demonstrating that probabilistic boolean formulae are not associative, while simultaneously providing insight into the relationship between the probability with which the underlying deterministic boolean formula is evaluated correctly and the structure of the probabilistic boolean formula. As a side-effect, we also demonstrate that certain types of probabilistic boolean formula satisfy the probability threshold conjecture.

To study the associativity of probabilistic boolean formulae, we define a probabilistic coloring experiment $\mathcal{E}$ as follows. Consider a balanced binary tree $G = (V, E)$, where V is the set of vertices and $E \subseteq (V \times V)$ a set of directed edges. Let $|V| = 2^{n+1} - 1$ for any positive integer n. Furthermore, let a directed path exist from the root vertex to any leaf vertex. We shall refer to this tree as a tree of size $2^{n+1} - 1$ and height n. For any directed edge $(u, v) \in E$, we will refer to v as the "child of" u and u as the "parent of" v. Clearly, every non-leaf vertex has two children and the root vertex has no parent. Let the $\frac{1}{2} < p \leq 1$ be the "probability parameter" associated with every vertex of G. The vertices of tree is colored red or blue according to the following rules. For any non leaf vertex v,

(1) If both its children are colored red, v is colored red with a probability p, and blue with a probability $1 - p$.

(2) If at least one of its children is colored blue, v is colored blue with a probability p and red with probability $1 - p$.

LEMMA 3.2.1. *If every leaf vertex of G is colored blue, the probability that the root vertex of G is colored red by $\mathcal{E}$ is at most $\sum_{i=1}^{n} q^i$, where $q = 1 - p$.*

PROOF. Henceforth, for the purposes of this proof, we consider only balanced binary directed trees, such that a directed path exists from the root vertex to any leaf vertex. Furthermore, all leaf vertices are colored blue and each of the vertices have a probability parameter p.

Consider a tree of size 3. This tree has a height 1. The probability that the root vertex of this tree is colored red by $\mathcal{E}$ is $f_1 = q = 1 - p$ (from the definition of $\mathcal{E}$).

For a tree of height 2 where the probability parameter of each of the vertices is p, the probability that the root is colored red by $\mathcal{E}$ is given by $q(1-f_1)^2 + 2q(1-f_1)f_1 + (1-q)f_1^2$. In general, if f_k is the probability that root of tree of height $k > 1$ is colored red by $\mathcal{E}$, $f_k = q(1-f_{k-1})^2 + 2q(1-f_{k-1})f_{k-1} + (1-q)f_{k-1}^2$.

Consider a tree of height 2. Then

$$\begin{aligned}
f_2 &= q(1-f_1)^2 + 2q(1-f_1)f_1 + (1-q)f_1^2 \\
&= q(1-f_1)(1-f_1+2f_1) + (1-q)f_1^2 \\
&= q(1-f_1^2) + f_1^2 - qf_1^2 \\
&= q - 2qf_1^2 + f_1^2 \\
&= q + f_1^2(1-2q) \\
&< q + f_1^2(1-q) && \text{since } 0 \le q \le 1 \\
&= q + q^2(1-q) && \text{since } f_1 = q \\
&< q + q^2 && \text{since } 0 \le q \le 1
\end{aligned}$$

In general, if $f_k < q + q^2 + q^3 + \cdots + q^k$,

$$\begin{aligned}
f_{k+1} &< q + f_k^2(1-q) \\
&< q + (q + q^2 + q^3 + \cdots + q^k)(q + q^2 + q^3 + \cdots + q^k)(1-q) \\
&= q + (q + q^2 + q^3 + \cdots + q^k)(q - q^{k+1}) \\
&< q + q^2 + q^3 + \cdots + q^{k+1} && \text{since } 0 \le q^{k+1} \le 1
\end{aligned}$$

Hence $f_n < \sum_{i=1}^{n} q^i$. Therefore the root vertex of a tree of height n where the probability parameter of each of the vertices is p, whose leaves are colored blue, is colored red by the probabilistic coloring experiment $\mathcal{E}$ with a probability at most $\sum_{i=1}^{n} q^i$, where $q = 1 - p$. $\square$

LEMMA 3.2.2. *If probability parameter of each of the vertices of G is $p = (1 - \frac{1}{2^c})$ and any leaf vertex of G is colored red with a probability $\hat{p} > \frac{1}{\sqrt{2}}$, (and blue with a probability $1 - \hat{p}$), the probability that the root* *vertex of* G *is colored red by* $\mathcal{E}$ *is at most*

$$\left(\frac{1}{2^{c-1}} - \frac{1}{2^{c(n-c')}}\right)$$

where c *is some positive real constant and* $c' = \lceil -\log(\log(\frac{1}{\hat{p}}))\rceil$

Proof. Henceforth, for the purposes of this proof, we consider only balanced binary directed trees, such that a directed path exists from any leaf vertex to the root vertex.

Consider a tree of height n. Let $q = (1-p)$. If f_k is the probability that the root vertex of a tree of height k is colored red by $\mathcal{E}$,

$$\begin{aligned} f_{k+1} &= q(1-f_k)^2 + 2q(1-f_k)f_k + (1-q)f_k^2 \\ &= q + f_k^2(1-2q) \end{aligned}$$

whenever $f_k > \frac{1}{\sqrt{2}}$, $q < 2qf_k^2$ and hence $q - 2qf_k^2 < 0$. Hence whenever $f_k > \frac{1}{\sqrt{2}}$, $f_{k+1} \le f_k^2$

Since $f_0 = \hat{p}$ (the leaves, trivially are trees of height 0 and are colored red with a probability $\hat{p}$), $f_1 < f_0^2$. Consider min c' such that, $f_{c'} < \frac{1}{2}$.

$$\hat{p}^{2^{c'}} < \frac{1}{2} \quad (3)$$

$$2^{c'}\log(\hat{p}) < -1 \quad (4)$$

$$2^{c'} < \frac{1}{\log\left(\frac{1}{\hat{p}}\right)} \quad (5)$$

$$c' = \left\lceil -\log\left(\log\left(\frac{1}{\hat{p}}\right)\right)\right\rceil \quad (6)$$

Let w be the root vertex of G and its two children be u, v respectively. Let us define the leaf vertices to be at level 0 of the tree, the vertices adjacent to the leaves to be at level 1 of the tree and so on. Consider the color of any vertex at level c' of the tree. It is red with a probability at most $\frac{1}{2}$ from Equation 6.

Consider the case when the color of the first consecutive $2^{n-c'-1}$ vertices (the vertices of the sub-tree rooted at u) is red and the color of the rest of the $2^{n-c'-1}$ vertices is blue. This is the case when the root is colored blue with the least probability (see appendix). From Lemma 3.2.1, u is colored red by $\mathcal{E}$ with a probability at most $\sum_{i=1}^{n-c'-1} q^i$. Hence, w is colored red by $\mathcal{E}$ with a probability

$$
\begin{aligned}
f_n &\le (1-q)\left(\sum_{i=1}^{n-c'-1} q^i\right) + q\left(1 - \sum_{i=1}^{n-c'-1} q^i\right) \\
&= (1-q)\left(\frac{q-q^{n-c'}}{1-q}\right) + q\left(1-\left[\frac{q-q^{n-c'}}{1-q}\right]\right) \\
&= q - q^{n-c'} + q - \left[\frac{q^2 - q^{n-c'+1}}{1-q}\right] \\
&\quad \text{for } n > c'+1 \text{ and } c > 1 \text{ we have} \\
f_n &\le 2q - q^{n-c'}
\end{aligned}
$$

Since $q = \frac{1}{2^c}$, we have $f_n \le \frac{1}{2^{c-1}} - \frac{1}{2^{c(n-c')}}$ where $c' = \left\lceil -\log\left(\log\left(\frac{1}{\hat{p}}\right)\right)\right\rceil$ □

Consider a tree $H = (V, E)$, where V is the set of vertices and $E \subseteq (V \times V)$ a set of directed edges. Let $|V| = 2n+1$ for some positive integer n. Let the vertices be labeled $v_1, v_2, v_3, \cdots, v_{2n+1}$. For $1 < i \le n$, let the vertex v_i have two children labeled v_{i-1} and v_{n+i+1} respectively and for v_1, the children be v_{n+1}, v_{n+2}. In H, a directed path exists from the root vertex to any leaf vertex, each non-leaf vertex has two children and both children of exactly one vertex are leaves. Furthermore $n-1$ vertices have one leaf and one non-leaf vertex as children. The height of such a tree is n and we shall refer to such trees as a "linear" tree. We shall consider a coloring experiment $\mathcal{E}$ as described above. Then

Lemma 3.2.3. *If probability parameter of each of the vertices of H is $p = (1 - \frac{1}{2^c})$ and any leaf vertex of H is colored red with a probability $\hat{p}$ (and blue with a probability $1-\hat{p}$), the probability that the root vertex of H is colored red by $\mathcal{E}$ is at least*

$$\frac{1}{2^c(1-\hat{p}) + 3\hat{p}}\left[1 + \frac{(2^c-3)^k(1-\hat{p})}{2^{\left(\frac{k+1}{c'} + c(k-1)\right)}}\right]$$

$$\frac{1}{2^{c-3} + 2\hat{p}} + \frac{1}{2^{\left(\frac{n+1}{c'}\right)}}$$

where c is some positive real constant and $c' = \log_{\frac{1}{\hat{p}}}(2)$.

Proof. For the purposes of this proof, all trees are considered to be linear trees, whose leaves are colored red with a probability $\hat{p}$.

Consider a linear tree of height 1. Let $q = 1-p$. The two leaves of this tree be colored red with a probability $\hat{p}$. The probability that the root vertex of this tree is colored red by $\mathcal{E}$ is $f_1 = q + (\hat{p})^2(2p-1)$. In general, if f_k is the probability that the root vertex of a tree of height k is colored red, then

$$\begin{aligned} f_{k+1} &= q + \hat{p} f_k (2p-1) \\ &> q + \hat{p} f_k (1-3q) \end{aligned}$$

If $r = \hat{p}(1-3q)$,

$$\begin{aligned} f_1 &> q + \hat{p}r \\ f_2 &> q + f_1 r \\ &= q + qr + \hat{p}r^2 \\ f_3 &> q + f_2 r \\ &= q + qr + qr^2 + \hat{p}r^3 \\ & \text{and in general} \\ f_k &> q + qr + qr^2 + \cdots + qr^{k-1} + \hat{p}r^k \\ &= q\left(\frac{1-r^k}{1-r}\right) + \hat{p}r^k \end{aligned}$$

Let $c' = \log_{\frac{1}{\hat{p}}}(2)$

$$\begin{aligned} f_k &> \frac{q}{1-r}(1-r^k) + \hat{p}r^k \\ &= \frac{q}{1-\hat{p}+3q\hat{p}}\left(1 - [\hat{p}]^k [1-3q]^k\right) + (\hat{p})^{k+1}(1-3q)^k \\ &= \frac{1}{2^c(1-\hat{p})+3\hat{p}}\left(1 - [\hat{p}]^k \left[1-\frac{3}{2^c}\right]^k + (2^c(1-\hat{p})+3\hat{p})\,[\hat{p}]^{k+1}\left[1-\frac{3}{2^c}\right]^k\right) \\ &> \frac{1}{2^c(1-\hat{p})+3\hat{p}}\left(1 + (2^c(1-\hat{p}))\,[\hat{p}]^{k+1}\left[1-\frac{3}{2^c}\right]^k\right) \\ &= \frac{1}{2^c(1-\hat{p})+3\hat{p}}\left[1 + \frac{(2^c-3)^k(1-\hat{p})}{2^{\left(\frac{k+1}{c'}+c(k-1)\right)}}\right] \end{aligned}$$

and hence, for a linear tree of height n where each of the vertices are associated with a probability parameter $p = 1 - \frac{1}{2^c}$ and the leaf vertices are colored red with a probability $\hat{p}$ (and blue with a probability $1-\hat{p}$), the probability that the root vertex is colored red by $\mathcal{E}$ is at least

$$\frac{1}{2^c(1-\hat{p})+3\hat{p}}\left[1 + \frac{(2^c-3)^n(1-\hat{p})}{2^{\left(\frac{n+1}{c'}+c(n-1)\right)}}\right]$$

Similarly, if $t = \hat{p}(1-2q)$,

$$\begin{aligned} f_k &= \frac{q}{1-t}(1-t^k) + \hat{p}t^k \\ &= \frac{\left(\frac{1}{2^c}\right)}{1-\hat{p}\left[1-\frac{2}{2^c}\right]}\left(1-[\hat{p}]^k\left[1-\frac{2}{2^c}\right]^k\right) + (\hat{p})^{k+1}\left(1-\frac{2}{2^c}\right)^k \\ &= \frac{1}{2^c(1-\hat{p})+2\hat{p}}\left(1-[\hat{p}]^k\left[1-\frac{2}{2^c}\right]^k\right) + (\hat{p})^{k+1}\left(1-\frac{2}{2^c}\right)^k \\ &< \frac{1}{2^c(1-\hat{p})+2\hat{p}} + \frac{1}{2^{\left(\frac{k+1}{c'}\right)}} \end{aligned}$$

and hence, for a linear tree of height n where each of the vertices are associated with a probability parameter $p = 1 - \frac{1}{2^c}$ and the leaf vertices are colored red with a probability $\frac{7}{8}$ (and blue with a probability $\frac{1}{8}$), the probability that the root vertex is colored red by $\mathcal{E}$ is at most

$$\frac{1}{2^c(1-\hat{p})+2\hat{p}} + \frac{1}{2^{\left(\frac{n+1}{c'}\right)}}$$

□

Lemma 3.2.2 and Lemma 3.2.3 show that the probability that the root vertex is colored red by the probabilistic coloring experiment is different for a balanced binary tree and a linear tree, which have the same number of leaf vertices (each of which are colored red with a probability $\hat{p}$). In both of these trees, the probability parameter of each vertex is $1 - \frac{1}{2^c}$. In fact a general inequality can be derived, where the probability parameters of the vertices are different.

Consider an tree $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ a set of directed edges. Furthermore, let a directed path exist from any leaf vertex to the root vertex. We had considered a probabilistic coloring experiment, where the probability parameter of each vertex of the tree was equal to p. We may consider a variant of the experiment where the probability parameter of a vertex v_i is p_i. Then the probabilistic coloring experiment $\mathcal{E}$ may be redefined as follows.

(1) The leaf vertices of G are colored red with a probability $r < MIN(p_j) : v_j \in V$.

(2) If v_i is a non leaf vertex, and both its children are colored red, v_i is colored red with a probability p_i and blue with a probability $q_i = 1 - p_i$.

(3) If v_i is a non leaf vertex, and at least one of its children is colored blue, v_i is colored blue with a probability p_i (and red with probability $q_i = 1 - p_i$).

Without loss of generality, let v_k be the root vertex of G and v_1 be a leaf vertex. Let the directed path from v_k to v_1 be $(v_k, v_{k-1}), (v_{k-1}, v_{k-2}), \cdots, (v_2, v_1)$. Furthermore, if v_i is any non leaf vertex in the path, let the children of v_i be v_{i-1} and $\hat{v}_{i-1}$. Consider a graph $H = (V, \hat{E})$ where for a particular $2 < j \le k$,

$\hat{E} = E - \{(v_2, v_1), (v_j, \hat{v}_{j-1})\} \cup \{(v_2, \hat{v}_{j-1}), (v_j, v_1)\}$. Informally, H is obtained by attaching a leaf vertex to v_j and the sub-tree rooted at $\hat{v}_{j-1}$ to v_2. We shall refer to H as the graph produced by a linearizing transformation of G.

LEMMA 3.2.4. *If H is the tree obtained by a linearizing transformation of G, the probability that the root vertex of H is colored red by $\mathcal{E}$ is at least equal to the probability that the root vertex of G is colored red by $\mathcal{E}$.*

PROOF. Initially, let $j = k$ (that is, let $v_j = v_k$, the root vertex). For any vertex $\hat{v}_i$ let f_i denote the probability that the vertex is colored red by $\mathcal{E}$ and let f_G be the probability that the root vertex of G is colored red by $\mathcal{E}$. From Lemma 3.2.1, it is easy to see that the probability that the root vertex of G is colored red by $\mathcal{E}$ is

$$\begin{aligned} f_G &= r \prod_{i=1}^{i=k-1} f_i(1-2q_{i+1}) \\ &+ q_2 \prod_{i=2}^{i=k-1} f_i(1-2q_{i+1}) \\ &+ q_3 \prod_{i=3}^{i=k-1} f_i(1-2q_{i+1}) \\ &+ \cdots \\ &+ q_{k-1} f_{k-1}(1-2q_k) \\ &+ q_k \end{aligned}$$

Similarly for H

$$\begin{aligned} f_H &= r(1-2q_k) f_{k-1} \prod_{i=1}^{i=k-2} f_i(1-2q_{i+1}) \\ &+ r(1-2q_k) q_2 \prod_{i=2}^{i=k-2} f_i(1-2q_{i+1}) \\ &+ \cdots \\ &+ r(1-2q_k) q_{k-2} f_{k-2}(1-2q_{k-1}) \\ &+ r(1-2q_k) q_{k-1} \\ &+ q_k \end{aligned}$$

And hence,

$$\begin{aligned} f_H - f_G &= (r - f_{k-1})(1-2q_k)q_2 \prod_{i=2}^{i=k-2} f_i(1-2q_{i+1}) \\ &+ (r - f_{k-1})(1-2q_k)q_3 \prod_{i=3}^{i=k-2} f_i(1-2q_{i+1}) \\ &+ \cdots \\ &+ (r - f_{k-1})(1-2q_k)q_{k-1} \end{aligned}$$

$f_H - f_G \geq 0$ since each of the product terms is positive and $r \leq f_{k-1}$. Let $j \neq k$, then for some intermediate node v_j, the probability of v_j being colored red by $\mathcal{E}$ in H is greater than equal to the probability of v_j being colored red by $\mathcal{E}$ in G by the above argument. Hence, again, $f_H \geq f_G$. □

Claim 3.2.1. *If G is a balanced binary tree with n leaves, there exists a sequence of linearizing transformations on G which produces H a linear tree with n leaves.*

Proof. The proof outline is by induction. Consider a balanced binary tree $G = (V, E)$ with 4 leaves (and height 2). If the two children of the root v_0 are labeled v_1, v_2 and their children are labeled v_3, v_4 and v_5, v_6 respectively, the linearizing transformation $H = (V, \hat{E})$ where $\hat{E} = E - \{(v_1, v_3), (v_0, v_2)\} \cup \{(v_1, v_2), (v_0, v_3)\}$ is the linearizing transformation which yields a linear tree H from G.

In general, assume that a sequence linearizing transformations which yields a linear tree exists for a balanced binary tree of height k. consider G, a tree of height $k+1$ with v_0 as the root vertex. Let v_1, v_2 be the two children of the root node. The sub-trees rooted at v_1, v_2 are of height k and hence, for each sub-tree, a sequence of linearizing transformations exist, which results in a linear tree rooted at v_1 and v_2 respectively. Let these sequences be S_1, S_2 respectively. Let $H' = (V, E')$ be the tree obtained by applying S_1 on v_1. Furthermore, let v_k be a vertex in the linear sub-tree rooted in v_1 such that v_k has two leaf children $v_{k+1}, \hat{v}_k$. Let $\hat{H} = (V, \hat{E})$ where $\hat{E} = E' - \{(v_k, v_{k+1}), (v_0, v_2)\} \cup \{(v_0, v_{k+1}), (v_k, v_2)\}$. Applying the sequence of linearizing transformations on the balanced binary sub-tree rooted at v_2 in $\hat{H}$ yields a linear tree. Hence a sequence of linearizing transformations on G yielding a linear tree exists. □

Theorem 3.3. *Probabilistic boolean formulae are not associative*

Proof. We will construct two probabilistic boolean formulae, F and F', in disjunctive normal form over m variables with identical clauses but different associations. We will demonstrate that for a fixed assignment of $I = 1^m$ to each variable in F and F', $F(I) \not\equiv F'(I)$ and hence $F \not\equiv F'$.

Consider a set of m boolean variables and their negations. We pick 3 variables from this set (with replacement) to form a deterministic conjunction. This will be referred to "picking a conjunctive clause of size 3 uniformly at random from a set of m variables". The probability that such a clause is satisfied by the assignment 1^m is $\frac{1}{8}$.

Pick n (where $n = 2^l$ for some positive integer l) conjunctive clauses of size 3, uniformly at random from a set of m variables. Let these clauses be labeled $C_1, C_2, \cdots, C_n$. For some $p = 1 - \frac{1}{2^c}$ where c is some positive real constant, construct the probabilistic boolean formula $F \equiv G_n$ as follows. Let $G_2 = (C_1 \vee_p C_2)$ and $G_i = (G_{i-1} \vee_p C_i)$ for $2 < i \leq n$. Correspondingly, construct $F' \equiv G'_1$ as follows. For $1 \leq i < 2^l$, $G'_i = (G'_{2i} \vee_p G'_{2i+1})$ and for $2^l \leq i < 2^l + n$, $G'_i \equiv C_{i-2^l+1}$. F and F' are in disjunctive normal form, with identical clauses but different associations.

For a fixed assignment 1^m, if $\textsc{unsat}_F$ is the probability that F is not satisfied,

$$\textsc{unsat}_F > \frac{1}{2^{c-3} + 2.625} \left[1 + \frac{(2^c - 3)^{n-1}}{2^{\left(\frac{n}{c'} + c(n-2) + 3\right)}} \right]$$

where $c' \approx 5.2$. This is from Lemma 3.2.3 and the observations

—A linear tree with n leaves has a height $n - 1$

—The boolean constant 0 corresponds to red and 1 corresponds to blue

—The leaves of the linear tree corresponds to the clauses $C_1, C_2, \cdots, C_n$. The probability that a leaf in the linear tree is colored blue $= \frac{1}{8} =$ probability that a size 3 conjunctive clause picked at random from m variables is satisfied by the assignment 1^m.

—The non leaf vertices correspond the operator $\vee_p$ and rules of the probabilistic coloring experiment $\mathcal{E}$ corresponds to the rules for determining the output of the operator $\vee_p$.

—$\log_{\frac{8}{7}}(2) \approx 5.2$ and $\frac{3 \times 7}{8} = 2.625$

Furthermore, for a fixed assignment 1^m, if $\textsc{unsat}_{F'}$ is the probability that F' is not satisfied,

$$\textsc{unsat}_{F'} < \frac{1}{2^{c-1}} - \frac{1}{2^{c(\log_2(n) - 3)}}$$

. This is from Lemma 3.2.2 and the observations

—A balanced binary tree with n leaves has a height $\log_2(n)$

—The boolean constant 0 corresponds to red and 1 corresponds to blue

—The leaves of the balanced tree corresponds to the clauses $C_1, C_2, \cdots, C_n$. The probability that a leaf in the balanced binary tree is colored blue $= \frac{1}{8} =$ probability that a size 3 conjunctive clause picked at random from m variables is satisfied by the assignment 1^m.

—The non leaf vertices correspond the operator $\vee_p$ and rules of the probabilistic coloring experiment $\mathcal{E}$ correspond to the rules for determining the output of the operator $\vee_p$.

—$\lceil -\log(\log(\frac{8}{7}))\rceil = 3$

Hence $\text{UNSAT}_{F'} < \text{UNSAT}_F$ and hence for an assignment of $I = 1^m$ for the variables in F and F', $F(I) \not\equiv F'(I)$ and hence $F' \not\equiv F$ and hence probabilistic boolean formulae are not associative. □

3.3 Satisfiability

Deterministic boolean formulae exhibit the following well studied satisfiability threshold phenomenon. Consider m variables and pick $m\alpha$ disjunctive clauses of size 3, uniformly at random from the set of m variables (as described in Section 3.2). Consider a $3-$CNF formula constructed by conjunction of these clauses. It is conjectured that a sharp threshold of satisfiability exists at around $\alpha^* \approx 4.2$ where whenever $\alpha > \alpha^*$, a random $3-$CNF formula is satisfiable with a low probability (or alternately, a random $3-$DNF formula constructed by picking $m\alpha$ disjunctive clauses of size 3, uniformly at random from a set of m variables, is satisfiable with a high probability whenever $\alpha > \alpha^*$). The satisfiability threshold conjecture has been studied through a variety of techniques, by deriving tail bounds for occupancy for example [Kamath et al. 1995]. These type of threshold phenomena have been studied under the general framework of $0-1$ laws [Kolaitis and Vardi 2000] as well.

Here, we demonstrate that the satisfiability threshold conjecture is likely to apply for certain types of probabilistic boolean formulae as well. Consider m variables and and pick $m\alpha = n = 2^l$ (for some positive integer l) conjunctive clauses of size 3, uniformly at random from the set of m variables. Consider a probabilistic disjunction of these clauses in some associative order, where the probability parameter of each probabilistic disjunction operator is $p > 1 - \frac{1}{2^c}$ for some positive real c.

Theorem 3.4. *The probability that F is satisfiable is high when $\alpha > 5.2$.*

Proof. We shall prove that the probability that F is satisfiable is at least

$$1 - \frac{1}{2^{c-3} + 1.75} - \frac{1}{2^{(\frac{n}{c'})}}$$

where $c' \approx 5.2$. Consider m variables and and pick $m\alpha = n = 2^l$ (for some positive integer l) conjunctive clauses of size 3, uniformly at random from the set of m variables. Let these clauses be labeled $C_1, C_2, \cdots, C_n$. For some $p > 1 - \frac{1}{2^c}$ where c is some positive real constant, construct the probabilistic boolean formula $F' \equiv G_n$ as follows. Let $G_2 = (C_1 \vee_p C_2)$ and $G_i = (G_{i-1} \vee_p C_i)$ for $2 < i \leq n$. For a fixed assignment 1^m, if $\text{SAT}_{F'}$ is the probability that F' is satisfied,

$$\text{SAT}_{F'} > 1 - \frac{1}{2^{c-3} + 1.75} - \frac{1}{2^{(\frac{n}{c'})}}$$

Where $c' = \log_{\frac{8}{7}} 2 \approx 5.2$ This is from Lemma 3.2.3 and the observations,

—A linear tree with n leaves has a height $n-1$

—The boolean constant 0 corresponds to red and 1 corresponds to blue

—The leaves of the linear tree corresponds to the clauses $C_1, C_2, \cdots, C_n$. The probability that a leaf in the linear tree is colored blue $= \frac{1}{8} =$ probability that a size 3 conjunctive clause picked at random from m variables is satisfied by the assignment 1^m.

—The non leaf vertices correspond the operator $\vee_p$ and rules of the probabilistic coloring experiment $\mathcal{E}$ corresponds to the rules for determining the output of the operator $\vee_p$.

Furthermore from Lemma 3.2.4 and Claim 3.2.1, $\text{SAT}_F \geq \text{SAT}_{F'}$. Hence, the probability that F is satisfied is at least $1 - \frac{1}{2^{c-3}+1.75} - \frac{1}{2^{\left(\frac{m\alpha}{c'}\right)}}$ and is high when $\alpha > 5.2$. $\square$

3.4 Probabilistic Boolean Identities

In this section, we define the equivalence of sample space generators and derive boolean identities. The equivalence of probabilistic boolean formulae are established based on the equivalence of their corresponding sample space generators. This notion of equivalence is not only useful in discovering identities, but also for the re-writing of probabilistic boolean formulae based on certain cost considerations (number of operators, range of probability parameters etc). These transformations will later play an important part in the synthesis of probabilistic boolean circuits.

Consider two probabilistic boolean formulae F, G and their sample space generators $\mathcal{S}_F$ and $\mathcal{S}_G$ respectively. $\mathcal{S}_F$ is equivalent to $\mathcal{S}_G$, denoted by $\mathcal{S}_F \equiv \mathcal{S}_G$, if the following conditions hold

—Either $\text{VAR}_{\mathcal{S}_F} \subseteq \text{VAR}_{\mathcal{S}_G}$ or $\text{VAR}_{\mathcal{S}_G} \subseteq \text{VAR}_{\mathcal{S}_F}$

—For any input I, $\mathcal{S}_{F(I)} \equiv \mathcal{S}_{G(I)}$

Intuitively, two probabilistic boolean formulae F, G are equivalent if, for every input $I \in \{0,1\}^k$, the random variable x whose sample space is $\mathcal{S}_{F(I)}$ is equivalent to the random variable y who sample space is $\mathcal{S}_{G(I)}$. Hence, two probabilistic boolean formulae are equivalent if their sample space generators are equivalent. Using the notion of equivalence of sample space generators, many probabilistic boolean identities may be derived. A few of the identities are enumerated below

Check these

—Operations with 0 and 1

$$\neg_p 0 \equiv \neg_{(1-p)} 1$$

$$\neg_p 1 \equiv \neg_{(1-p)} 0$$

$$0 \wedge_p x \equiv \neg_{(1-p)} 0$$

$$0 \vee_p x \equiv \neg_{(1-p)} x$$

$$1 \wedge_p x \equiv \neg_{(1-p)} x$$

$$1 \vee_p x \equiv \neg_{(1-p)} 1$$

—Probabilistic Tautology

$$x \vee_p x \equiv \neg_{(1-p)} x$$

$$x \wedge_p x \equiv \neg_{(1-p)} x$$

—Laws of Complementation

$$x \vee_p (\neg x) \equiv \neg_{(1-p)} 1$$

$$x \wedge_p (\neg x) \equiv \neg_{(1-p)} 0$$

—Probabilistic Absorption

$$x \vee_p (x \wedge y) \equiv (x \vee_p x) \equiv \neg_{(1-p)} x$$

$$x \wedge_p (x \vee y) \equiv (x \wedge_p x) \equiv \neg_{(1-p)} x$$

—Distributive Law

$$x \wedge_p (y \vee z) \equiv (x \wedge y) \vee_p (x \wedge z)$$

$$x \vee_p (y \wedge z) \equiv (x \vee y) \wedge_p (x \vee z)$$

—Probabilistic DeMorgan Law

$$\neg_p (x \vee y) \equiv \neg (x \vee_p y) \equiv (\neg x) \wedge_p (\neg y)$$

$$\neg_p (x \wedge y) \equiv \neg (x \wedge_p y) \equiv (\neg x) \vee_p (\neg y)$$

—Probabilistic DeMorgan Law - General case

$$\neg_p (x \vee_q y) \equiv \neg_{pq+(1-p)(1-q)} (x \vee y) \equiv \neg (x \vee_{pq+(1-p)(1-q)} y) \equiv (\neg x) \wedge_{pq+(1-p)(1-q)} (\neg y)$$

$$\neg_p (x \wedge_q y) \equiv \neg_{pq+(1-p)(1-q)} (x \wedge y) \equiv \neg (x \wedge_{pq+(1-p)(1-q)} y) \equiv (\neg x) \vee_{pq+(1-p)(1-q)} (\neg y)$$

4. RELATIONSHIP BETWEEN PROBABILISTIC BOOLEAN FORMULAE, SAMPLE SPACE GENERATORS AND PROBABILISTIC BOOLEAN FUNCTIONS

In this section, we explore the relationships between probabilistic boolean formulae, probabilistic boolean functions and sample space generators. We show that every probabilistic boolean formula computes a probabilistic boolean function and for any given probabilistic boolean function there exists a probabilistic boolean formula that computes it.

4.1 Probabilistic Boolean Functions

Informally, a $k-$ bit probabilistic boolean function is a function whose domain is $\{0,1\}^k$ and for any input $I \in \{0,1\}^k$, the output of the function is a boolean random variable x_I. Formally, A boolean random variable is defined to be a random variable with two possible values 0 and 1. Any boolean random variable x is associated with a sample space composed of two types of sample points. Those associated with the event $x = 0$ and those associated with the event $x = 1$. Two random variables are said to be equivalent if their sample spaces are equivalent.

Definition 4.1. A $k-$ bit probabilistic boolean function $\mathcal{B}$ is a function whose domain is the set of $k-$ bit binary strings $\{0,1\}^k$ and whose range is a set of boolean random variables.

Two probabilistic boolean functions $\mathcal{B}$ and $\mathcal{B}'$ are said to be equivalent if their domains are identical and for any input I, the random variable $\mathcal{B}(I)$ is equivalent to the random variable $\mathcal{B}'(I)$.

The notion of a sample space generator as defined in Section 2.1 is independent of probabilistic boolean formulae. If the cardinality of the set of boolean variables in a sample space generator $\mathcal{S}$ is k, then the sample space generator defines a $k-$ bit probabilistic boolean function $\mathcal{B}$. For any input $I \in \{0,1\}^k$ to $\mathcal{B}$, the random variable $x_I \equiv \mathcal{B}(I)$ is characterized by the sample space generated by $\mathcal{S}$ with an input I. Consider a PBF F and its sample space generator $\mathcal{S}_F$. The probabilistic boolean function $\mathcal{B}_F$ defined by $\mathcal{S}_F$ will be referred to as "the probabilistic boolean function computed by F" or interchangeably as "the probabilistic boolean function of F". Since there is a sample space generator associated with every PBF, every PBF computes a probabilistic boolean function.

Consider a $k-$ bit probabilistic boolean function $\mathcal{B}$ such that there is a distinguished $i \in \{0,1\}^k$ such that for any input $j \in \{0,1\}^k$,

$$\mathcal{B}(j) = \begin{cases} x & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

where x is a boolean random variable. We will call such a probabilistic boolean function as a $k-$ bit probabilistic boolean impulse function at i. Now

LEMMA 4.1.1. *For every $k-$ bit probabilistic boolean impulse function, there exists a probabilistic boolean formula, which computes that function*

PROOF. Consider a $k-$ bit probabilistic boolean impulse function $\mathcal{B}$ at i such that $\mathcal{B}(i) = x$ where $x = 1$ with probability p and $x = 0$ with probability $(1-p)$. Since given any (deterministic) boolean function, there exists a (deterministic) boolean formula that computes it, there exists a (deterministic) boolean formula G such that the boolean function it computes, $\mathcal{G}$ is of the form

$$\mathcal{G}(j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

Consider $F \equiv ((\neg_p 0) \wedge G)$. F is the probabilistic boolean formula which computes $\mathcal{B}$. □

THEOREM 4.2. *For every probabilistic boolean function $\mathcal{B}$ there exists a probabilistic boolean formula F such that F computes $\mathcal{B}$.*

PROOF. Let $\mathcal{B}$ be a $k-$ bit probabilistic boolean function.

For any $i, j \in \{0,1\}^k$, let $\mathcal{B}_i$ be a probabilistic boolean impulse function such that ,

$$\mathcal{B}_i(j) = \begin{cases} \mathcal{B}(j) & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

From lemma 4.1.1, there exists a probabilistic boolean formula F_i which computes $\mathcal{B}_i$. Since

$$\mathcal{B} \equiv \vee_{i \in \{0,1\}^k} \ \mathcal{B}_i$$

$$F \equiv \vee_{i \in \{0,1\}^k} \ F_i$$

is a probabilistic boolean formula which computes $\mathcal{B}$. □

Summarizing this section, we know that

(1) There is a sample space generator associated with every probabilistic boolean formula

(2) Every sample space generator defines a probabilistic boolean function

(3) Every probabilistic boolean formula computes a probabilistic boolean function

(4) For any probabilistic boolean function $\mathcal{B}$ there exists a probabilistic boolean formula F which computes it.

(5) For any probabilistic boolean function $\mathcal{B}$ there exists a sample space generator $\mathcal{S}$ which defines it.

5. PROBABILISTIC BOOLEAN CIRCUITS AND RELATIONSHIP WITH CLASSICAL MODELS OF COMPUTATION

Some applications, notably those which implement probabilistic algorithms, signal processing algorithms etc, are resilient in the presence of "errors" in their computing steps [Chakrapani et al. 2007; George et al. 2006]. These steps may be characterized as probabilistic boolean functions, and "erroneous" circuits maybe utilized to implement these steps. As mentioned in Section 1, probabilistic boolean formulae may be utilized to model such circuits with "erroneous" gates. We shall call these circuits with "erroneous" gates as *probabilistic boolean circuits*. The rest of this section will be devoted to defining and characterizing probabilistic boolean circuits, and relating them to probabilistic boolean formulae, probabilistic boolean functions, sample space generators and other classical models in computer science such as randomized circuits and probabilistic automata.

Analogous to conventional boolean circuits, probabilistic boolean circuits may be defined as follows. A probabilistic boolean circuit is a directed acyclic connected graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is a set of directed edges. The vertices are of three kinds. Input vertices, of in-degree 0 and out degree 1 associated with boolean variables (called input variables of the circuit) or boolean constants $\{0, 1\}$, internal vertices associated with one of three operators $\vee_p, \wedge_p, \neg_p$ where $0 < p \leq 1$. Vertices associated $\vee_p$ and $\wedge_p$ have in-degree 2 and out-degree 1, those associated with $\neg_p$ have in-degree and out-degree 1. There is one-distinguished output vertex of in-degree 1 and out-degree 0. For any assignment of boolean values 0 and 1 to the input variables of the circuit, the value of the input vertex is the boolean value assigned to the corresponding boolean variable, or the boolean constant associated with the vertex. The value of an edge $(u, v) \in E$ is the value associated with the vertex u. The value of any internal vertex, is the value obtained by applying the probabilistic boolean operator associated with the vertex, to values associated with the incoming edges. Finally, the value *computed* by the probabilistic boolean circuit is the value associated with the output vertex.

Claim 5.0.1. *For any probabilistic boolean formula F and the probabilistic boolean function $\mathcal{B}$ it computes, there exists a probabilistic boolean circuit $\mathbb{C}_F$ which computes an equivalent probabilistic boolean function.*

This claim is straightforward, a valid PBF is obtained by recursive application of the rules outlined in Section 2. An equivalent probabilistic boolean circuit may be constructed by creating input vertices for every boolean variable and constant in the PBF and an internal vertex for every boolean operator.

Claim 5.0.2. *For every probabilistic boolean function $\mathcal{B}$ there exists a probabilistic boolean circuit $\mathbb{C}$ such that $\mathbb{C}$ computes $\mathcal{B}$.*

This claim can be proved from Theorem 4.2 which proves that for every probabilistic boolean function $\mathcal{B}$ there exists a probabilistic boolean formula that computes $\mathcal{B}$ and Claim 5.0.1.

5.1 Relationship with Randomized Boolean Circuits

In computer science literature, randomized boolean circuits have been used as a model to study randomized algorithms [Sipser 1997]. Analogous to conventional boolean circuits, a randomized boolean circuit is a directed acyclic connected graph $G = (V, E)$, with three types of vertices, input vertices of in-degree 0 and out degree 1 associated with boolean variables (called input variables of the circuit), boolean constants or boolean random variables, internal vertices associated with one of three boolean operators $\vee, \wedge, \neg$. Vertices associated $\vee$ and $\wedge$ have in-degree 2 and out-degree 1 those associated with $\neg$ have in-degree and out-degree 1. Any internal vertex $v \in V$ has the property that there is at most one edge (u, v) such that $u \in V$ is an input vertex associated with a boolean random variable. There is one-distinguished output vertex of in-degree 1 and out-degree 0. For any assignment of boolean values 0 and 1 to the input variables of the circuit, the value of the input vertex is the boolean value assigned to the corresponding boolean variable, the boolean constant associated with the vertex or the output of random experiment associated with the corresponding boolean random variable. The value of an edge (u, v) is the value associated with the vertex u. The value of any internal vertex is the valued obtained by applying the boolean operator associated with the vertex to values of the incoming edges. Finally, the value *computed* by the randomized boolean circuit is the value associated with the output vertex.

Claim 5.1.1. *For every probabilistic boolean function $\mathcal{B}$, there exists a randomized boolean circuit which computes $\mathcal{B}$.*

This claim can be proved by a trivial modification of Lemma 4.1.1, and Theorem 4.2 while noting that the input vertices of a randomized circuit may be associated with boolean random variables. Since every randomized circuit computes a probabilistic boolean function, and for any given probabilistic boolean function, there exists a probabilistic boolean circuit that computes it, trivially there exists an equivalent probabilistic boolean circuit for every randomized circuits. However, in the sequel, we outline a procedure to construct such a circuit.

5.2 Relationship with Probabilistic Automata

So far, we have considered only those probabilistic boolean circuits whose operators have a probability of correctness, p, where $p \in \mathbb{Q}$. For the purposes of the sequel, we consider $p \in \mathbb{R}$ and $0 \leq p \leq 1$. Rabin, in his seminal paper introduced Probabilistic automata [Rabin 1963]. Following Rabin's classic paper [Rabin

1963], a probabilistic automaton over an alphabet Σ is a system $< S, M, s_0, Q >$ where $S = \{s_0, \cdots, s_n\}$ is a finite set (of states), M is a function from $S \times \Sigma$ into $[0,1]^{n+1}$ (the transition probabilities table) such that for $(s, \sigma) \in S \times \Sigma$, the transition function $M(s, \sigma) = (p_0(s, \sigma), \cdots, p_n(s, \sigma))$ where $0 \leq p_i(s, \sigma)$ and $\sum p_i(s, \sigma) = 1$. $s_0 \in S$ is the initial state and $F \subseteq S$ is the set of designated final states. Rabin has shown that a probabilistic automata with transition probabilities and *cut points* in the real interval $[0, 1]$ are strictly more powerful than deterministic finite automata and those with rational cut points in the interval $[0, 1]$ are equivalent to deterministic finite automata.

The transition function of probabilistic automata may be simulated using probabilistic as well randomized boolean circuits. Without loss of generality, let the input alphabet Σ of the probabilistic automaton be the set $\{0, 1\}$ and the set of states be encoded in binary. Given an input $\sigma \in \Sigma$ and a current state $s \in S$, $0 \leq i, j \leq n$ such that $p_i, p_j \neq 0$ (and hence $p_i + p_j = 1$. That is, given any state and an input, the set of possible next states (with non zero transition probability) is at most of cardinality two. Without loss of generality let $i < j$, now the transition function M may be redefined as a probabilistic boolean function $M : \Sigma \times S \rightarrow \{0, 1\}$, where the output 0 indicates the next state is s_i and output 1 indicates that the next state is s_j. This probabilistic boolean function may be computed by a probabilistic (or randomized) boolean circuit.

6. ENERGY IMPLICATIONS OF PROBABILISTIC BOOLEAN CIRCUITS

So far, we have defined and characterized probabilistic boolean circuits as a model of computation and related them to randomized circuits. In addition, we wish to differentiate probabilistic boolean circuits and randomized circuits along two critical dimensions. First, as we shall see below, for every randomized circuit of a particular depth and size, there exists a probabilistic boolean circuit of identical depth and size, which computes an equivalent probabilistic boolean function. However, because of thermodynamic reasons, in CMOS based realizations, an "inherently" probabilistic gate (and hence a probabilistic circuit) consumes lesser energy than a randomized circuit realized in the form of a deterministic gate with a random input bit [Korkmaz et al. 2006]. Earlier, this has been demonstrated experimentally for a certain class of applications [Korkmaz et al. 2006; Chakrapani et al. 2006; George et al. 2006; Chakrapani et al. 2007]. This is complementary to a prior result, which showed that probabilistic algorithms consume lesser energy than their deterministic counterparts (even though they may be identical in classical time complexity measure) in the BRAM model of computation [Palem 2003]. Secondly, such a separation in energy consumption is like to be enhanced when we take into account the energy cost of producing a random bit. This is expanded upon in the sequel.

First, we shall outline a construction to derive a probabilistic boolean circuit of identical depth, size and functionality from a randomized boolean circuit. Consider a randomized boolean circuit $\mathbb{C}$. Consider the set of input vertices U associated with boolean random variables in $\mathbb{C}$. Consider any $u \in U$ and an internal vertex v such that $(u, v), (v, w) \in \mathbb{C}$ (we ignore trivial circuits where v is the output vertex). Let u be associated with boolean random variable x_u such that probability that $x_u = 1$ is p_u. Consider the internal vertex v. We note that at most one incoming edge of v is incident on an input vertex associated with a boolean random variable (from definition). v is associated with the boolean operator $\vee, \wedge$ or $\neg$. Replace v with v' where v' is associated correspondingly with the boolean operator $\vee_p, \wedge_p$ or $\neg_p$. Replace u with a vertex u' such that u' is associated with boolean constant 1. This new circuit will be denoted by $\mathbb{C}/\{u\}$.

Lemma 6.0.1. *For any assignment of boolean constants $\{0, 1\}$ to the boolean variables of $\mathbb{C}$ and a corresponding assignment to the boolean variables of $\mathbb{C}/\{u\}$, the boolean random variable m which characterizes the value of the edge (v, w) is equivalent to the boolean random variable m' which characterizes the value of (v', w).*

Proof. Let v be associated with $\vee$ or $\wedge$ (the proof for the case where v is associated with $\neg$ is along the same lines) and let the two incoming edges of v in $\mathbb{C}$ be (u, v) and (t, v) (and correspondingly the incoming edges of v' in $\mathbb{C}/u$ be (u', v') and (t, v')). Since $\mathbb{C}/u$ and $\mathbb{C}$ are identical except for vertices u, v, u' and v', the boolean random variable which characterizes the value of (t, v) is equivalent to the boolean random variable which characterizes the value of (t, v'). Let this variable be n. Assume $m \not\equiv m'$. We consider the following cases

—v is associated with $\wedge$. Then it must be the case that $n \wedge x_u \not\equiv n \wedge_{p_u} 1$. However, this is a contradiction

—v is associated with $\vee$. Then it must be the case that $n \vee x_u \not\equiv n \vee_{p_u} 1$. However, this is a contradiction

□

Theorem 6.1. *For every randomized boolean circuit $\mathbb{C}$ there exists a probabilistic boolean circuit $\hat{\mathbb{C}}$ such that $\mathbb{C}$ and $\hat{\mathbb{C}}$ compute equivalent probabilistic boolean functions. Furthermore $\mathbb{C}$ and $\hat{\mathbb{C}}$ are isomorphic.*

Proof. outline: Consider a randomized boolean circuit $\mathbb{C}$. Let U be the set of input vertices associated with boolean random variables in $\mathbb{C}$. For any $u \in U$, from Lemma 6.0.1 we know that $\mathbb{C}$ and $\mathbb{C}/\{u\}$ compute equivalent probabilistic boolean functions. Through simple induction, it can be shown that $\mathbb{C}$ and $\mathbb{C}/U$ compute equivalent probabilistic boolean functions. $\mathbb{C}/U$ is the probabilistic boolean circuit $\hat{\mathbb{C}}$ and is isomorphic it $\mathbb{C}$ □

From Theorem 6.1 it is evident that for every randomized boolean circuit $\mathbb{C}$ there exists and isomorphic probabilistic boolean circuit $\hat{\mathbb{C}}$ such that $\mathbb{C}$ and $\hat{\mathbb{C}}$ compute equivalent probabilistic boolean functions. Furthermore, under this isomorphism, the set of input vertices U associated with boolean random variables in $\mathbb{C}$, is a subset of the set of input vertices of $\hat{\mathbb{C}}$ associated with boolean constants. Furthermore, let $v \in \mathbb{C}$ be an internal vertex then the isomorphic vertex $\hat{v} \in \hat{\mathbb{C}}$ is an internal vertex as well. Moreover, if v is associated with $\vee, \wedge$ or $\neg$, $\hat{v}$ is associated with $\vee_p, \wedge_p$ or $\neg_p$ correspondingly.

Through thermodynamic arguments, analytical models and measurements, it can be demonstrated that in the domain of CMOS, there is a relationship between the energy consumption of a gate and its probability of correctness; or equivalently, "inherently" probabilistic gates consume lesser energy than their deterministic counterparts. Figure 0**??** illustrates the relationship between the energy consumption and probability of correctness for an inverter synthesized using TSMC 0.25 micron technology. The various curves in the figure plot the simulated (using HSpice) energy consumption, the analytically modeled energy consumption and the actual energy consumption of an inverter realized using this technology. Figure 0**??** illustrate the relationship between energy consumption and probabilities of correctness for gates which implement other boolean operators. Hence, in CMOS based implementation of probabilistic and randomized boolean, the gate which implements the operator associated with $\hat{v}$ consumes substantially lesser energy than that which implements the operator associated with v. Hence, depending on the proportion of the vertices which are associated with probabilistic boolean operators, the energy consumption of a probabilistic boolean circuit is likely to be substantially lesser than an equivalent randomized boolean circuit.

In the study of randomized algorithms and randomized circuits, randomness has rightly been identified as a resource. This consideration has given rise to a rich body of work, which seeks to address this concern through several techniques, from recycling of random bits [Impagliazzo and Zuckerman 1989], to techniques which extract randomness from weak random sources [Chor and Goldreich 1985] and methods to "amplify" randomness through pseudo-random number generators [Blum and Micali 1984]. An interested reader is referred to the huge body of work in this area, and an overview is beyond the scope of this paper. However, another critical consideration is the energy required to generate random bits, either through extraction from weak random sources or through pseudo-random number generators. If this energy is taken into account, probabilistic boolean circuits implemented in CMOS which are composed of "inherently" probabilistic gates, are likely to be more energy efficient than randomized circuits—whose functionality critically depend on the availability of random bits—when the cost of producing random bits are taken into account. This separation in energy consumption is in addition to the thermodynamic energy efficiency obtained due to implementing probabilistic boolean operators in CMOS.

An experimental demonstration of the energy efficiency of probabilistic boolean circuits when compared to randomized circuits is described in [Chakrapani et al. 2006; Chakrapani et al. 2007] and is briefly illustrated in Table 0**??** below. This work accounts for the separation in energy consumption due to the energy cost of producing random bits, as well as the energy efficiency gained by implementing probabilistic boolean operators directly in CMOS in the form of "inherently" probabilistic gates [Cheemalavagu et al. 2005]. Hence

THEOREM 6.2. *For every randomized boolean circuit $\mathbb{C}$ there exists a probabilistic boolean circuit $\hat{\mathbb{C}}$ such that $\mathbb{C}$ and $\hat{\mathbb{C}}$ compute equivalent probabilistic boolean functions. Furthermore $\hat{\mathbb{C}}$ consumes lesser energy than $\mathbb{C}$ in* CMOS *based implementations.*

7. CONCLUSION AND FUTURE DIRECTIONS

In this work, we have introduced *probabilistic boolean algebra* as an area of study, derived some important properties and related it to classical models from computer science. We have also shown that in practical CMOS implementations, probabilistic boolean circuits are likely to consume substantially lesser energy than randomized circuits. Our future inquiry is along three directions (i) synthesis of a probabilistic boolean circuit which computes a given probabilistic boolean function $\mathcal{B}$: Our technique involves constructing a sample space generator $\mathcal{S}$ which defines $\mathcal{B}$, optimizing $\mathcal{S}$ based on some cost criteria and constructing a probabilistic boolean formula F from $\mathcal{S}$ and will be elaborated elsewhere. (ii) Fast parallel evaluation of circuit reliability in probabilistic boolean circuits: Rabin [Rabin 1963] has demonstrated how acceptance probabilities of input strings (for a given probabilistic automaton) may be computed by setting "cut-points" and constructing corresponding deterministic automata. This technique may be extended to probabilistic boolean circuits to evaluate their reliability for specific inputs.

REFERENCES

BAHAR, R. I., MUNDY, J., AND CHEN, J. 2003. A probabilistic-based design methodology for nanoscale computation. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*. 480–486.

BLUM, M. AND MICALI, S. 1984. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput. 13,* 4, 850–864.

CHAITIN, G. J. AND SCHWARTZ, J. T. 1978. A note on monte carlo primality tests and algorithmic information the ory. *Communications on Pure and Applied Mathematics 31*, 521–527.

CHAKRAPANI, L. N., AKGUL, B. E. S., CHEEMALAVAGU, S., KORKMAZ, P., PALEM, K. V., AND SESHASAYEE, B. 2006. Ultra efficient embedded soc architectures based on probabilistic cmos technology. In *Proceedings of The 9th Design Automation and Test in Europe (DATE)*. 1110–1115.

CHAKRAPANI, L. N. B., KORKMAZ, P., AKGUL, B. E. S., AND PALEM, K. V. 2007. Probabilistic system-on-a-chip architectures. *To appear in ACM Transactions on Design Automation of Electronic Systems (ACM-TODAES).*

Cheemalavagu, S., Korkmaz, P., Palem, K. V., Akgul, B. E. S., and Chakrapani, L. N. 2005. A probabilistic CMOS switch and its realization by exploiting noise. In *Proceedings of the IFIP international conference on very large scale integration*.

Chor, B. and Goldreich, O. 1985. Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In *IEEE Symposium on Foundations of Computer Science*. 429–442.

Davis, M. 2001. *Engines of Logic: Mathematicians and the Origin of the Computer*. W. W. Norton and Company.

George, J., Marr, B., Akgul, B. E. S., and Palem, K. 2006. Probabilistic arithmetic and energy efficient embedded signal processing. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems CASES*.

Impagliazzo, R. and Zuckerman, D. 1989. How to recycle random bits. In *IEEE Symposium on Foundations of Computer Science*. 248–253.

itrs. 2002. International technology roadmap for semiconductors 2002 update.

Kamath, A., Motwani, R., Palem, K. V., and Spirakis, P. G. 1995. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Struct. Algorithms 7,* 1, 59–80.

Kolaitis, P. G. and Vardi, M. Y. 2000. 0-1 laws for fragments of existential second-order logic: A survey. *Mathematical Foundations of Computer Science*, 84–98.

Korkmaz, P., Akgul, B. E. S., Chakrapani, L. N., and Palem, K. V. 2006. Advocating noise as an agent for ultra low-energy computing: Probabilistic cmos devices and their characteristics. *Special Issue of Japanese Journal of Applied Physics (JJAP) 45,* 4B (Apr.), 3307–3316.

Moore's Law. http://www.intel.com/technology/silicon/mooreslaw/.

Motwani, R. and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press.

Nepal, K., Bahar, R. I., Mundy, J., Patterson, W. R., and Zaslavsky, A. 2005. Designing logic circuits for probabilistic computation in the presence of noise. In *Proceedings of the 42nd Design Automation Conference*. 485–490.

Palem, K. V. 2003. Proof as experiment: Probabilistic algorithms from a thermodynamic perspective. In *Proceedings of The International Symposium on Verification (Theory and Practice),*. Taormina, Sicily.

Palem, K. V. 2005. Energy aware computing through probabilistic switching: A study of limits. *IEEE Transactions on Computers 54,* 9, 1123–1137.

Pippenger, N. 1985. On networks of noisy gates. *Proceedings of the 26th Annual IEEE Symposim on Foundations of Computer Science*, 30–38.

Pippenger, N. 1989. Invariance of complexity measures for networks with unreliable gates. *Journal of the ACM 36*, 531–539.

Pippenger, N., Stamoulis, G. D., and Tsitsiklis, J. N. 1991. On a lower bound for the redundancy of reliable networks with noisy gates. *IEEE Transactions on Information Theory 37,* 3, 639–643.

Rabin, M. O. 1963. Probabilistic automata. *Information and Control 6*, 230–245.

Rabin, M. O. 1976. Probabilistic algorithms. In *Algorithms and Complexity, New Directions and Recent Trends*, J. F. Traub, Ed. 29–39.

Sipser, M. 1997. *Introduction to the Theory of Computation*. PWS Publishing Company.

von Neumann, J. 1956. Probabilistic logics and the synthesis of reliable organizms from unreliable components. *Automata Studies*, 43–98.