# Algorithms for Generating Probabilities with Multivalued Stochastic Relay Circuits

David T. Lee and Jehoshua Bruck, *Fellow, IEEE*

**Abstract—**The problem of random number generation dates back to Von Neumann's work in 1951. Since then, many algorithms have been developed for generating unbiased bits from complex correlated sources as well as for generating arbitrary distributions from unbiased bits. An equally interesting, but less studied aspect is the *structural* component of random number generation. That is, given a set of primitive sources of randomness, and given composition rules induced by a device or nature, how can we build networks that generate arbitrary probability distributions? In this paper, we study the generation of arbitrary probability distributions in multivalued relay circuits, a generalization in which relays can take on any of $N$ states and the logical 'and' and 'or' are replaced with 'min' and 'max' respectively. These circuits can be thought of as modeling the timing of events which depend on other event occurences. We describe a duality property and give algorithms that synthesize arbitrary rational probability distributions. We prove that these networks are robust to errors and design a universal probability generator which takes input bits and outputs any desired binary probability distribution.

**Index Terms—**Multiple valued logics, random number generation, stochastic relays

## 1 Introduction

### 1.1 Why Study Stochasticity?

Many biological systems exhibit stochasticity. Examples of these include gene expression [3], chemical reactions [18], and neuron signaling [13]. However, despite the stochasticity, often deemed as noise, they still achieve functionalities that artificial systems cannot compete with. A natural question arises: Is stochasticity a bug in nature or an important tool? It is known that, in the case of randomized algorithms [9], stochasticity is the key to more powerful information processing and computation.

Motivated by this, we tackle a fundamental question as a stepping stone: Can we design networks that can generate any desired probability distribution? An understanding of this would be essential for the further study of probabilistic computation.

### 1.2 Random Number Generation Given Structural Constraints

The problem of generating probability distributions exists in an important thread of work in computer science. In 1951, Von Neumann [19] studied this problem in the context of generating a fair coin toss from a biased coin. Knuth and Yao [7] then studied the reverse problem of generating arbitrary distributions from unbiased bits (fair coins). This work was extended by Han and Hoshi [6] and Abrahams [1] to generating arbitrary distributions from biased distributions.

- *D. T. Lee is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305. E-mail: davidtlee@stanford.edu.*
- *J. Bruck is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125. E-mail: bruck@caltech.edu.*

*Manuscript received 4 Apr. 2013; revised 30 May 2014; accepted 22 Jan. 2015. Date of publication 5 Feb. 2015; date of current version 11 Nov. 2015. Recommended for acceptance by A. Nannarelli.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TC.2015.2401027*

In physical systems, however, randomness arises in specific components that can only be composed together according to given rules. We cannot use algorithms that assign output symbols to arbitrary coin tossing sequences. Rather, certain structural constraints determine how physical building blocks can be composed. In essence, the problem of random number generation needs to be studied from the perspective of physical devices and network synthesis.

In 1962, Gill [5] studied the design of deterministic sequential machines that could transform a random input source to an arbitrary random output source. Sheng [17] considered probability transformers with threshold logic elements. Recently, Wilhelm and Bruck [21] studied the synthesis of relay circuits when given stochastic relays; Zhou et al. [23] studied the synthesis of flow networks given stochastic splitters; and Qian et al. [11] studied the use of combinatorial logic circuits to transform random bit streams, and showed that one could use probability distributions to represent information.

In this paper, we will analyze random number generation in the context of multivalued relay circuits, a generalization of standard relay circuits to any number of states. This generalization is a simple abstraction of event-based networks (such as neural networks) in which one event is triggered upon the activation of some Boolean function of other event activations (see Section 9). While stochasticity is known to play a role in neural networks, it has not yet been understood what this role is.

### 1.3 Deterministic Relays

We will start by introducing deterministic relays. A deterministic relay switch is a two terminal object which can be connected (closed) by a wire or left open. The state of the switch, which can be either 0 or 1, describes an open or closed state respectively (see Fig. 1a). These states are complements of each other. That is, if switch $x$ is in state 0, then the complement $\bar{x}$ is in state 1 and vice versa.

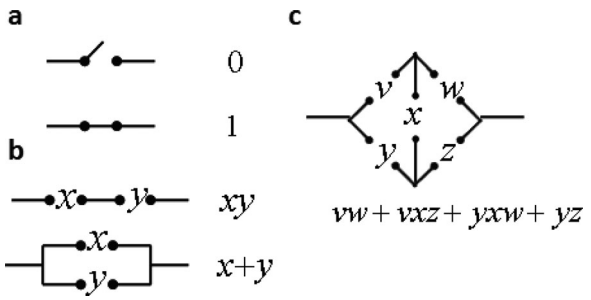



Fig. 1. *Deterministic examples.* For two states, $xy$ is the Boolean 'and' of $x$ and $y$ while $x + y$ is the Boolean 'or'. Figures b and c also apply to multivalued relays where $xy$ is $\min(x, y)$ and $x + y$ is $\max(x, y)$. (a) A relay can either be opened (state 0) or closed (state 1). (b) In series, the entire circuit can only be closed if both $x$ and $y$ are closed. In parallel, the entire circuit will be closed if either $x$ or $y$ are closed. (c) In this example of a non-sp circuit, we take the Boolean 'and' along the four possible paths, and then apply the Boolean 'or' to those subsequent values.

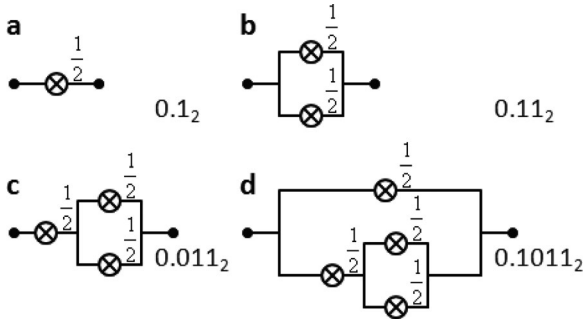



Fig. 2. *Simple examples.* As stated, $\frac{1}{2}$ is short for $(\frac{1}{2}, \frac{1}{2})$. (a) A single $\frac{1}{2}$-pswitch. (b) Putting (a) in parallel with a $\frac{1}{2}$-pswitch gives $\frac{3}{4}$. (c) Putting (b) in series with a $\frac{1}{2}$-pswitch gives $\frac{3}{8}$. (d) Putting (c) in parallel with a $\frac{1}{2}$-pswitch gives $\frac{11}{16}$.

When multiple switches are composed together, these networks are known as relay circuits. One of these, a series composition, is formed when two switches are put end to end. A parallel composition is formed when two switches are composed so that the beginning terminals are connected together and the end terminals are connected together. Relay circuits are defined to be in the closed state 1 if there exists a closed path from the beginning to end terminal and 0 otherwise.

Shannon showed that series and parallel compositions can be represented by the Boolean 'and' and 'or' operations [15]. If switches $x$ and $y$ are composed in series to form $z_1$, then $z_1$ will only be closed if both $x$ and $y$ are closed. If switches $x$ and $y$ are composed in parallel to form $z_2$, then $z_2$ will be closed if either $x$ or $y$ are closed. Circuits formed solely by series and parallel compositions are called sp circuits. We will denote the series composition of $x$ and $y$ by $x * y$ or simply $xy$ and the parallel composition by $x + y$ (see Fig. 1b). This notation will be preserved for further generalizations of the relay circuits.

Shannon showed that non sp circuits could also be represented by Boolean operations (see Fig. 1c). The general mathematical representation of any relay circuits would be to find all paths that go from the beginning to the end terminal. For each path, take the Boolean 'and' of all switches along that path; then take the Boolean 'or' of the values for each path.

### 1.4 Stochastic Relays

Recently, Wilhelm and Bruck introduced the notion of a stochastic relay circuit [21]. These circuits are a generalization of Shannon's relay circuits; instead of having deterministic relay switches that are in either the open or closed state, stochastic relay circuits can exist in either state with a specified probability. If a stochastic relay switch $x$, called a pswitch, has probability $p$ of being closed and probability $1 - p$ of being open, this *distribution* is represented by a vector $v = (1 - p, p)$ where $v_i$ corresponds to the probability of $x$ being in state $i$. We say that $x$ *realizes* $(1 - p, p)$ or simply $x$ *realizes* $p$. If pswitches $x$ and $y$, which realize probabilities $p$ and $q$ respectively, are composed in series, the new composition will realize $pq$. If they are composed in parallel, the new composition will realize $p + q - pq$ (see Fig. 2).

One of the primary questions dealt with in their work was the generation of probability distributions using a limited number of base switch types, known as a switch set. For example, relay circuits built with the switch set $\mathbf{S} = \{\frac{1}{2}\}$ can only use pswitches with the distribution $(\frac{1}{2}, \frac{1}{2})$. They proved that using the switch set $\mathbf{S} = \{\frac{1}{2}\}$, all probability distributions $\frac{a}{2^n}$ could be realized with at most $n$ pswitches, where $a$ is a non-negative integer. Continuing, many more results were proved not only in realizing other probability distributions [21], [22], but also in circuit constructions such as a universal probability generator [21] and in robustness [8] and expressibility [22] properties of these circuits.

### 1.5 Multivalued Stochastic Relays

In order to study non-Bernoulli random variables, it is necessary to generalize Shannon's relays to a larger number of states. Multivalued logics have been studied as early as in 1921 by Post [10] and followed up on by Webb [20] and others [12]. The work presented in this paper concerns one generalization of two-state relay circuits to multivalued relay circuits where we use two multivalued functions (gates).

A multivalued switch is a relay switch that can be in any of n states: 0, 1, 2, . . . , n − 1. We define the complement of a switch to be $n - 1 - i$, where $i$ is the state of the switch. Series and parallel compositions are redefined to 'min' and 'max', respectively, rather than the Boolean 'and' and 'or'. This means that when switches $x$ and $y$ are composed in series, the overall circuit is in state $\min(x, y)$ and when they are composed in parallel, the overall circuit is in state $\max(x, y)$ (see Fig. 3, examples a and b). Non-sp circuits are also defined in a similar way to two-state circuits. The general mathematical representation of any multivalued relay circuit is to find all paths that go from the beginning to the end terminal. For each path, we take the 'min' of all switches along that path; then we take the 'max' of the values derived for each path (Fig. 1, examples b and c still apply).

One physical understanding of this max-min algebra is found in the timing of relay switches. Let all switches start in the closed position and open at some time $t$. Then the state of the switch will be the time $t \in \{0, 1, \ldots, n - 1\}$ when the switch opens. Compose two switches, which open at time $t_1$ and $t_2$, in series. If either switch opens, then the overall connection is broken. Therefore, the time that the series composition will be open is $\min(t_1, t_2)$. In the same way, we

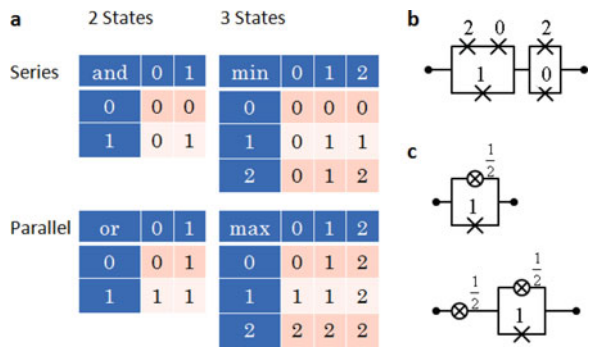



Fig. 3. *Three-state examples.* We use $\times$ to represent a deterministic switch and $\otimes$ to represent a pswitch. (a) A comparison of the truth tables for two-state and three-state relay switches. (b) A simple three-state deterministic example. Evaluating series connections using min and parallel connections as max, we find that this circuit is in state 1. (c) A simple three-state stochastic example. On the top, we put a $(\frac{1}{2}, 0, \frac{1}{2})$ pswitch in parallel with a deterministic switch in state 1. Then, with $\frac{1}{2}$ probability we get $\max(0, 1)$ and with $\frac{1}{2}$ probability we get $\max(2, 1)$, which yields a $(0, \frac{1}{2}, \frac{1}{2})$ circuit. Similarly, the bottom circuit realizes distribution $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

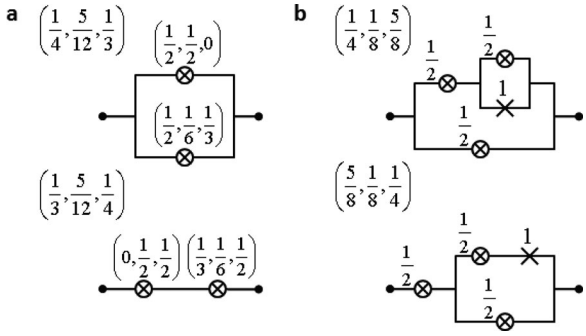



Fig. 4. *Three-state duality example.* Remember that the three-state $\frac{1}{2}$-pswitch is the shorthand for the $(\frac{1}{2}, 0, \frac{1}{2})$ pswitch. (a) A two element duality example. (b) A duality example on a larger circuit. Note that since the pswitches are symmetric distributions (duals of themselves), only the compositions are changed.

can compose the switches in parallel. Since both switches need to be opened in order for the overall circuit to be open, the time calculated for the parallel composition is $\max(t_1, t_2)$. More generally, the max-min algebra can be used to understand the timing of events when an event occurence depends on the occurence of other events, i.e., event $A$ occurs if event $B$ occurs or if events $C$ and $D$ occur. Max-min functions have been used to reason about discrete event systems [4].

Now that we have defined multivalued relays, we can also reason about probability distributions over the $n$ states of a multivalued stochastic relay. If a three-state pswitch $x$ has probability $p_0$ of being in state 0, $p_1$ of being in state 1, and $p_2$ of being in state 2, we represent the distribution with the vector $v = (p_0, p_1, p_2)$ and say $x$ realizes $(p_0, p_1, p_2)$. If the distribution is in the form $(1 - p, 0, \ldots, 0, p)$, we will shorten this to simply $p$ if the number of states can be inferred from the context or if the equation holds for any number of states (see Fig. 3c).

We present the following results in our paper:

1) A duality property (Section 2).
2) Networks for generating binary probability distributions (Sections 3 and 4).
3) Networks for generating rational distributions (Section 5).
4) Robustness of the previous networks to switch errors (Section 6).
5) Universal probability generation (Section 7).
6) Switching with partially ordered states (Section 8).
7) Discussion on applications (Section 9).

## 2 Duality

Duality is a well-known concept which plays a role in resistor networks, deterministic and two-state stochastic relay circuits. We start by characterizing duality in multivalued circuits: Define the dual state of $i$ as the state $n - 1 - i$; the dual distribution of $v$ as the distribution $\bar{v}$ where $\bar{v}_i = v_{n-1-i}$; the dual switch of $x$ as the switch $\bar{x}$ that realizes the dual distribution of $x$; and the dual circuit of $C$ as the circuit $\bar{C}$ that realizes the dual distribution of $C$.

**Theorem 1 (Duality Theorem).** *Given a stochastic series-parallel circuit $C$, we can construct $\bar{C}$ by replacing all the switches in $C$ with their dual switches and by replacing series connections with parallel connections and vice versa. (see Fig. 4).*

**Proof.** This is shown using induction on series-parallel connections.

Base case. The dual of a single pswitch with distribution $(p_0, p_1, \ldots, p_{n-1})$ is $(p_{n-1}, \ldots, p_1, p_0)$, which trivially satisfies the theorem.

Inductive step. Suppose a circuit $C$ with distribution $(p_0, p_1, \ldots, p_{n-1})$ and a circuit $C'$ with distribution $(q_0, q_1, \ldots, q_{n-1})$ satisfy the theorem, i.e., the distribution of $\bar{C}$ is $(p_{n-1}, \ldots, p_1, p_0)$ and the distribution of $\bar{C}'$ is $(q_{n-1}, \ldots, q_1, q_0)$. To prove the theorem, it is sufficient for us to show that $CC'$ is the dual of $\bar{C} + \bar{C}'$.

Let $C_s = CC'$ and $C_p = \bar{C} + \bar{C}'$. Then $C_s = (c_0, c_1, \ldots, c_{n-1})$ and $C_p = (d_0, d_1, \ldots, d_{n-1})$ where

$$\begin{aligned} c_k &= \sum_{\min(i,j)=k} Pr(C = i) Pr(C' = j) \\ &= \sum_{\min(i,j)=k} p_i q_j \\ d_k &= \sum_{\max(i,j)=k} Pr(\bar{C} = i) Pr(\bar{C}' = j) \\ &= \sum_{\max(i,j)=k} p_{n-1-i} q_{n-1-j} \\ &= \sum_{\min(n-1-i,n-1-j)=n-1-k} p_{n-1-i} q_{n-1-j} \\ &= \sum_{\min(i',j')=n-1-k} p_{i'} q_{j'} = c_{n-1-k}. \end{aligned}$$

We see that $d_0 = c_{n-1}$, $d_1 = c_{n-2}, \ldots, d_{n-1} = c_0$, demonstrating that $C_s$ is the dual of $C_p$. □

## 3 Realizing Binary Three-State Distributions

We can now ask questions about generating probability distributions with stochastic relay circuits. We will begin by demonstrating how to generate binary distributions on three states.

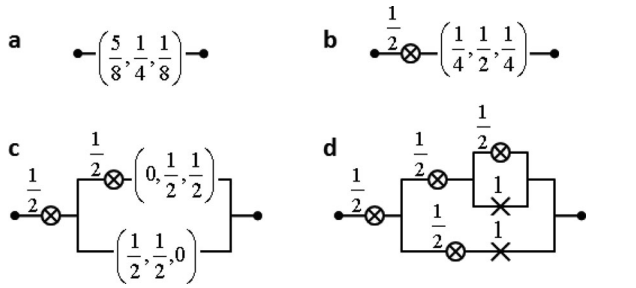



Fig. 5. *Realizing three-state distribution.* (a) We want to realize $(\frac{5}{8},\frac{1}{4},\frac{1}{8})$. This falls under the first case. (b) Now we have an unconstructed distribution that falls under the second case. (c) There are two unconstructed distributions. The top one falls under case 3 and the bottom one falls under case 2. (Note that case 2 is equivalent to case 1 for $p_0 = \frac{1}{2}$). (d) The final circuit.

In writing out algorithms and circuit constructions we will use the notation introduced earlier for series and parallel connections: $xy$ and $x + y$ will denote series and parallel connections respectively. We will use this notation loosely, i.e., $(\frac{1}{2},0,\frac{1}{2})(\frac{1}{2},\frac{1}{2},0)$ represents a circuit formed by composing a $(\frac{1}{2},0,\frac{1}{2})$ pswitch in series with a $(\frac{1}{2},\frac{1}{2},0)$ pswitch.

**Lemma 1 (Three-state sp composition rules).** *Given* $p = (p_0,p_1,p_2)$ *and* $q = (q_0,q_1,q_2)$, *let* $x = pq$ *and* $y = p + q$. *Then,*

$$\begin{aligned} x_0 &= p_0q_0 + p_0q_1 + p_0q_2 + p_1q_0 + p_2q_0 \\ &= 1 - (1-p_0)(1-q_0) \\ x_1 &= p_1q_1 + p_1q_2 + p_2q_1 \\ x_2 &= p_2q_2 \\ y_0 &= p_0q_0 \\ y_1 &= p_1q_1 + p_1q_0 + p_0q_1 \\ y_2 &= p_2q_2 + p_2q_1 + p_2q_0 + p_1q_2 + p_0q_2 \\ &= 1 - (1-p_2)(1-q_2). \end{aligned}$$

**Proof.** The above expressions follow from enumerating all $3^2$ switch combinations. □

**Theorem 2 (Binary three-state Distributions).** *Using the switch set* $\boldsymbol{S} = \{\frac{1}{2}\}$ *and the deterministic switches, we can realize all 3-state distributions of the form* $(\frac{a}{2^n},\frac{b}{2^n},\frac{c}{2^n})$ *using at most* $2n-1$ *pswitches with the recursive construction of Algorithm 1 (see Fig. 5 for an example).*

**Algorithm 1.** Binary three-state construction

**Input:** A probability distribution $p$ of the form $(\frac{a}{2^n},\frac{b}{2^n},\frac{c}{2^n})$
**Output:** A recursive constructive of $p$
**if** $p \in S$ **then**
  **return** $p$;
**else if** $\frac{a}{2^n} > \frac{1}{2}$ **then**
  **return** $(\frac{1}{2},0,\frac{1}{2})(\frac{a-2^{n-1}}{2^{n-1}},\frac{b}{2^{n-1}},\frac{c}{2^{n-1}})$;
**else if** $\frac{a+b}{2^n} > \frac{1}{2}$ **then**
  **return** $(\frac{a}{2^{n-1}},\frac{2^{n-1}-a}{2^{n-1}},0) + (\frac{1}{2},0,\frac{1}{2})(0,\frac{2^{n-1}-c}{2^{n-1}},\frac{c}{2^{n-1}})$;
**else**
  **return** $(\frac{a}{2^{n-1}},\frac{b}{2^{n-1}},\frac{c-2^{n-1}}{2^{n-1}}) + (\frac{1}{2},0,\frac{1}{2})$;
**end**

**Proof.** For any distribution $p = (p_0,p_1,p_2)$, there exists some smallest $k$ such that $\sum_{i=0}^{k} p_i > \frac{1}{2}$, which correspond to the three recursive cases enumerated above. We can verify that for each of these cases:

1) The decompositions obey the three-state composition rules.
2) The switches are valid (non-negative probabilities and sum to 1)

Since each algorithm's decomposition uses switches of type $(\frac{a}{2^{n-1}},\frac{b}{2^{n-1}},\frac{c}{2^{n-1}})$, then we will eventually have $n = 0$, corresponding to a deterministic switch; at this point the algorithm terminates and has successfully constructed any $(\frac{a}{2^n},\frac{b}{2^n},\frac{c}{2^n})$.

We will now prove that we use at most $2n-1$ pswitches for all $n \geq 1$. Define $f_n$ as the maximum number of pswitches used in the construction of any distribution $(\frac{a}{2^n},\frac{b}{2^n},\frac{c}{2^n})$. Then,

$$f_1 = 1 \tag{1}$$

$$\begin{aligned} f_n &= \max(f_{n-1}+1, 2(n-1)+1, f_{n-1}+1) \\ &= \max(f_{n-1}+1, 2n-1), \end{aligned} \tag{2}$$

where (1) is shown trivially, and (2) comes from the three cases of Algorithm 1. Note that we are using a previous result [21] that two state distributions of form $(\frac{a}{2^n},\frac{b}{2^n})$ use at most $n$ switches. We are now left with a simple induction exercise.

Base case. $f_1 = 1 = 2(1) - 1$.

Inductive step. Assume $f_k = 2k-1$. Then,

$$\begin{aligned} f_{k+1} &= \max(f_k+1, 2(k+1)-1) \\ &= \max(2k, 2(k+1)-1) \\ &= 2(k+1)-1. \end{aligned}$$

So $f_k = 2k-1 \Rightarrow f_{k+1} = 2(k+1)-1$. □

It is useful at this time to provide some intuition regarding the algorithm. We can view the original distribution as a series of blocks dividing up the interval $[0,1]$ (see Fig. 6). By applying the algorithm, we are separating this larger interval into smaller intervals $[0,\frac{1}{2}]$ and $[\frac{1}{2},1]$, cutting any

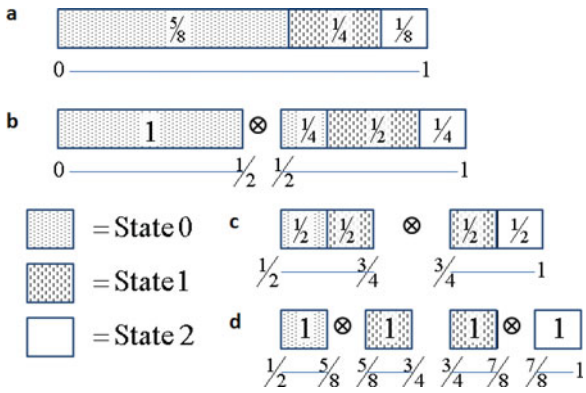



Fig. 6. *Block-interval visualization of three-state algorithm.* We want to realize $(\frac{5}{8},\frac{1}{4},\frac{1}{8})$. A $\otimes$ symbol marks where the cuts occur which correspond to the use of one pswitch. (a) We start with one interval split into blocks corresponding to the pswitch probabilities. (b) When we cut $[0,1]$ in half, we get two intervals; the blocks are cut and the probabilities change according to the new interval size. (c) The second application of the algorithm. (d) When we are left with single block-interval pairs, we know we are done since the deterministic switch is in our switch set.

blocks on that boundary. When this separation occurs, the total size of the block is decreased (namely, in half), and so the probabilities representing those intervals change—these probabilities are precisely those of the algorithm.

It is interesting to point out that this interpretation bears remarkable similarity to Han and Hoshi's interval algorithm [6]. The main difference is that we require a static circuit which cannot change dynamically based on the randomness created thus far. This structural constraint prevents a straightforward application of the interval algorithm.

## 4 Realizing Binary $N$-State Distributions

Intuitively, we can describe the algorithm for $N$ states in the same way as for three states. We first find the smallest index $k$ for which $\sum_{i=0}^{k} p_i > \frac{1}{2}$. Then based on the index $k$, we can decompose our distribution in a way corresponding to the interval-block visualization; the only difference is that each interval can now have up to $N$ block-types instead of just 3.

**Theorem 3 (Binary N-state Distributions).** *Using the switch set $\mathbf{S} = \{\frac{1}{2}\}$ and the deterministic switches, we can realize all $N$-state distributions of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$ with at most $f_{n,N} =$*

$$\begin{cases} 2^n - 1, & n \leq \lceil \log_2 N \rceil \\ 2^{\lceil \log_2 N \rceil} - 1 + (N-1)(n - \lceil \log_2 N \rceil), & n \geq \lceil \log_2 N \rceil, \end{cases}$$

*switches, using the recursive circuit construction of Algorithm 2 (see Fig. 7 for an example).*

**Algorithm 2.** Binary $N$-state construction. Note that we are using $\frac{1}{2}$ as a shorthand for $(\frac{1}{2}, 0, \ldots, 0, \frac{1}{2})$ as explained in the introduction.

**Input:** A probability distribution $p$ of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$
**Output:** A recursive construction of $p$
**if** $p \in S$ **then**
  **return** $p$;
**else if** $\frac{x_0}{2^n} > \frac{1}{2}$ **then**
  **return** $\frac{1}{2}(\frac{x_0 - 2^{n-1}}{2^{n-1}}, \frac{x_1}{2^{n-1}}, \ldots, \frac{x_{N-1}}{2^{n-1}})$;
**else if** $\frac{x_0 + x_1}{2^n} > \frac{1}{2}$ **then**
  **return** $(\frac{x_0}{2^{n-1}}, \frac{2^{n-1} - x_0}{2^{n-1}}, \ldots) + \frac{1}{2}(0, \frac{x_0 + x_1 - 2^{n-1}}{2^{n-1}}, \frac{x_2}{2^{n-1}}, \ldots)$;
**else if** ... **then**
  ...;
**else**
  **return** $(\frac{x_0}{2^{n-1}}, \frac{x_1}{2^{n-1}}, \ldots, \frac{x_{N-1} - 2^{n-1}}{2^{n-1}}) + \frac{1}{2}$;
**end**

**Proof.** Because the proof is primarily algebraic and similar in structure to that of the three state relays, we will detail it in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2015.2401027 available online. □

We briefly note that our bounds in Theorem 3 are tight when circuits are generated from Algorithm 2. That is, a probability distribution exists such that following Algorithm 2 will exactly achieve the upper bound. This follows from the fact that the inductive steps in our proof consisted entirely of equalities. However, it is still an open problem whether one can achieve better bounds by using circuits outside the range of the given algorithm.

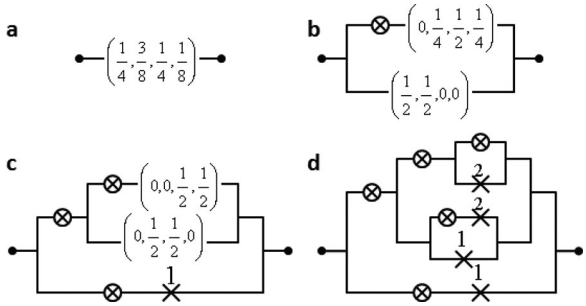



Fig. 7. *Realizing four-state distribution.* We use the $N$-state algorithm on the a four-state distribution. A $\otimes$ symbol represents the pswitch $(\frac{1}{2}, 0, 0, \frac{1}{2})$. (a) We want to realize $(\frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{8})$. Initially, $p_0 + p_1 > \frac{1}{2}$. (b) For the top distribution, $p_0 + p_1 + p_2 > \frac{1}{2}$. For the bottom distribution, $p_0 + p_1 > \frac{1}{2}$. (c) For the top distribution, $p_0 + p_1 + p_2 + p_3 > \frac{1}{2}$. For the bottom distribution, $p_0 + p_1 + p_2 > \frac{1}{2}$. (d) The final circuit.

## 5 Realizing Rational Distributions

Given the previous results, a natural question arises. Can we also generate probability distributions over non-binary fractions? More generally, can we generate distributions for any rational distribution? This question was studied for the two-state case by Wilhelm, Zhou, and Bruck [21], [22] for distributions of the form $(\frac{a}{q^n}, \frac{b}{q^n})$. In their work, they demonstrated algorithms for realizing these distributions for any $q$ that is a multiple of 2 or 3. In addition, they proved that for any prime $q > 3$, no algorithm exists that can generate all $(\frac{a}{q^n}, \frac{b}{q^n})$ using the switch set $\mathbf{S} = \{\frac{1}{q}, \frac{2}{q}, \ldots, \frac{q-1}{q}\}$.

We approach the question of realizing rational $N$-state distributions from two angles and demonstrate algorithms for each. In the first case, we show how to reduce the generation of any $N$ state distribution to the generation of several two-state distributions using only a $\frac{1}{2}$ switch. In the second case, we show another generalization of previous binary $N$ state results that allows us to generate arbitrary rational distributions, albeit with a larger switch set. The key for these algorithms lies in a generalization of the block-interval construction of binary distributions.

**Lemma 2 (Block-interval construction of distributions).** *Let $p = (p_0, p_1, \ldots, p_{N-1})$ be any distribution. Then given any 'cut', represented by the distribution $1 - q = (q, 0, \ldots, 0, 1 - q)$,*

$$\begin{aligned} &(p_0, p_1, \ldots, p_{N-1}) \\ &\quad = \left(\frac{p_0}{q}, \ldots, \frac{p_{k-1}}{q}, \frac{q - \sum_{i=0}^{k-1} p_i}{q}, 0, \ldots\right) \\ &\qquad + \left[(1-q)\left(\ldots, 0, \frac{\sum_{i=0}^{k} p_i - q}{1-q}, \frac{p_{k+1}}{1-q}, \ldots, \frac{p_{N-1}}{1-q}\right)\right] \\ &\quad = \left[\left(\frac{p_0}{q}, \ldots, \frac{p_{k-1}}{q}, \frac{q - \sum_{i=0}^{k-1} p_i}{q}, 0, \ldots\right) + (1-q)\right] \\ &\qquad * \left(\ldots, 0, \frac{\sum_{i=0}^{k} p_i - q}{1-q}, \frac{p_{k+1}}{1-q}, \ldots, \frac{p_{N-1}}{1-q}\right), \end{aligned}$$

*where $k$ is the index satisfying $q \geq \sum_{i=0}^{k-1} p_i$, $q \leq \sum_{i=0}^{k} p_i$,*

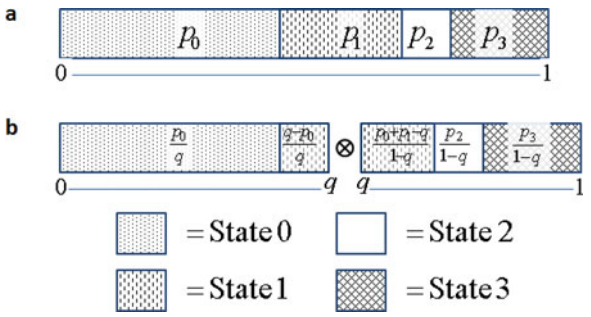



Fig. 8. *Block-interval visualization (rational).* We want to realize $(p_0, p_1, p_2, p_3)$. A $\otimes$ symbol marks where the cut occurs, which corresponds to the use of one pswitch. (a) We start with one interval split into blocks corresponding to the pswitch probabilities. (b) When we cut with pswitch $(q, \ldots, 1-q)$, the interval is separated into two intervals: $[0, q]$ and $[q, 1]$.

**Proof.** The equation follows directly from the max-min composition rules. □

Essentially, if a distribution is represented as blocks in an interval (Fig. 8), then any "cut" on this interval corresponds to a reduction of this distribution into a pair of distributions that can be represented by the resulting intervals.

### 5.1 State Reduction

Given the results already proved on two-state distributions, one natural way to tackle $N$ states is to first reduce the given $N$-state distribution into compositions of two-state distributions. Then, if algorithms already exist for those two-state forms, we are done.

**Theorem 4 (State Reduction Algorithm).** *Using the switch set $S = \{\frac{1}{2}\}$, we can reduce any N-state distribution of the form $(\frac{x_0}{q}, \frac{x_1}{q}, \ldots, \frac{x_{N-1}}{q})$ into at most $N-1$ two-state distributions of the form $(\ldots, \frac{x_i}{q}, \ldots, \frac{x_j}{q}, \ldots)$ using at most $f_{\log_2 q, N} = 2^{\lceil \log_2 N \rceil} - 1 + (N-1)(\log_2 q - \lceil \log_2 N \rceil)$ switches with the recursive construction of Algorithm 3 (for an example, see Fig. 9a).*

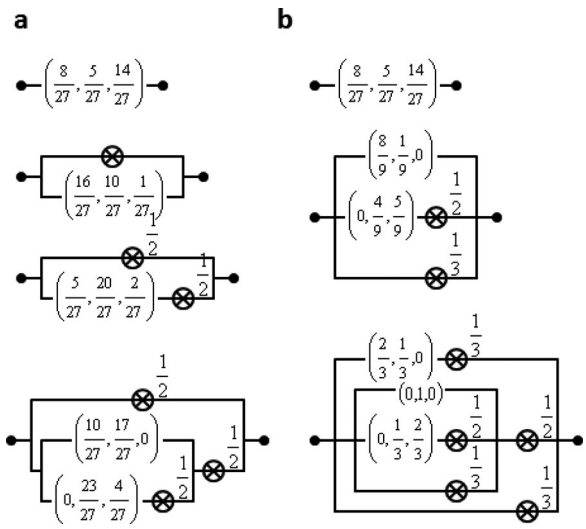



Fig. 9. *Algorithm illustrations.* (a) An example of the state recursion algorithm. (b) An example of the denominator recursion algorithm.

**Algorithm 3.** State reduction construction

**Input:** A probability distribution $p$ of the form $(\frac{x_0}{q}, \frac{x_1}{q}, \ldots, \frac{x_{N-1}}{q})$
**Output:** A recursive construction of $p$
**if** $p$ is a two state distribution **then**
  **return** $p$;
**else if** $\frac{x_0}{q} > \frac{1}{2}$ **then**
  **return** $\frac{1}{2}(\frac{2x_0-q}{q}, \frac{2x_1}{q}, \ldots, \frac{2x_{N-1}}{q})$;
**else if** $\frac{x_0+x_1}{q} > \frac{1}{2}$ **then**
  **return** $(\frac{2x_0}{q}, \frac{q-2x_0}{q}, \ldots) + \frac{1}{2}(0, \frac{2x_0+2x_1-q}{q}, \frac{2x_2}{q}, \ldots)$;
**else if** ... **then**
  ...;
**else**
  **return** $(\frac{2x_0}{q}, \frac{2x_1}{q}, \ldots, \frac{2x_{N-1}-q}{q}) + \frac{1}{2}$;
**end**

**Proof.** The proof is essentially identical to Theorem 3, so we will simply sketch it here. First, we know from Lemma 2 that the breakdown of probabilities in each round is valid. To prove the complexity, we note that we can simply replace $n$ (from Theorem 3) with $\log_2 q$ since we will have reduced all switches to at most two states at that point. □

### 5.2 Denominator Reduction

The second approach to realizing rational $N$-state distributions is to use a different switch set to directly reduce the power in the denominator. The intuition comes from a generalization of the block-interval construction. Rather than just cutting it at one point, i.e., at $\frac{1}{2}$ for realizing binary distributions, we can cut it at the points $\frac{1}{q}, \frac{2}{q}, \ldots,$ and $\frac{q-1}{q}$ to get $q$ equally sized intervals.

**Theorem 5 (Denominator Reduction Algorithm).** *Using $S = \{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{q}\}$ and the deterministic switches, we can realize any distribution $(\frac{x_0}{q^n}, \frac{x_1}{q^n}, \ldots, \frac{x_{N-1}}{q^n})$ using at most $q^{\lceil \log_q N \rceil} - 1 + (N-1)(q-1)(n - \lceil \log_q N \rceil)$ switches with the recursive construction of Algorithm 4 (for an example, see Fig. 9b).*

**Algorithm 4.** Denominator reduction construction

**Input:** A probability distribution $p$ of the form $(\frac{x_0}{q^n}, \frac{x_1}{q^n}, \ldots, \frac{x_{N-1}}{q^n})$
**Output:** A recursive construction of $p$
Define the following quantities for $j = 0, 1, 2, \ldots, q$;
$k_j$ = the smallest index at which $\sum_{i=0}^{k_j} p_i > \frac{j}{q}$;
$L_j = \frac{\sum_{i=0}^{k_j} x_i - jq^{n-1}}{q^{n-1}}$;
$R_j = \frac{(j+1)q^{n-1} - \sum_{i=0}^{k_{j+1}-1} x_i}{q^{n-1}}$;
**if** $p \in S$ **then**
  **return** $p$;
**else**
  **return** $(\frac{x_0}{q^{n-1}}, \ldots, \frac{x_{k_1-1}}{q^{n-1}}, R_0, 0, \ldots) + \frac{1}{2}(\cdots, 0, L_1, \frac{x_{k_1+1}}{q^{n-1}}, \ldots, \frac{x_{k_2-1}}{q^{n-1}}, R_1, 0, \ldots) + \cdots + \frac{1}{q-1}(\cdots, 0, L_{q-1}, \frac{x_{k_{q-2}+1}}{q^{n-1}}, \ldots, \frac{x_{k_{q-1}-1}}{q^{n-1}}, R_{q-1}, 0, \ldots) + \frac{1}{q}(\cdots, 0, L_q, \frac{x_{k_{q-1}+1}}{q^{n-1}}, \ldots, \frac{x_{N-1}}{q^{n-1}})$;
**end**

**Proof.** The idea of the construction is to perform $q-1$ iterations of the block-interval construction lemma for each reduction of a pswitch. WLOG, let the original interval

have length 1. Then the first cut at $\frac{q-1}{q}$ gives intervals of length $\frac{q-1}{q}$ and $\frac{1}{q}$. The second cut at $\frac{q-2}{q-1}$ of the $\frac{q-1}{q}$ interval gives intervals of length $\frac{q-1}{q}\frac{q-2}{q-1}=\frac{q-2}{q}$ and $\frac{q-1}{q}\frac{1}{q-1}=\frac{1}{q}$. The third cut leaves an interval of length $\frac{q-1}{q}\frac{q-2}{q-1}\frac{q-3}{q-2}=\frac{q-3}{q}$ and another $\frac{1}{q}$ interval. As this continues, we get $q$ intervals of length $\frac{1}{q}$. Our expression is exactly that of the block-interval construction lemma applied to the above description. Then since at each 'round' of these decompositions, we reduce the exponent of the denominator by 1, we will eventually terminate when $n=0$, which is a deterministic switch.

The proof for the complexity is similar to the $N$-state binary proof, so we will simply sketch it here. Let $f_{q,n,N}$ be the number of switches needed. Then we have the base cases of $f_{q,n,1}=0$ and $f_{q,0,N}=0$. Now, we know that $f_{q,n,N}=\max_{i_1,i_2,\ldots,i_q:\sum_j i_j=N+q-1}(q-1+\sum_j f_{q,n-1,i_j})$. Then assuming our expression holds for all $f_{q,\hat{n},N}$ such that $\hat{n}\leq n-1$, we can claim that the indices maximizing the given expression must be when all $i_j\in\{\lceil\frac{N+q+1}{q}\rceil, \lfloor\frac{N+q+1}{q}\rfloor\}$. Then we can simply verify that the resulting expression will evaluate to the closed form. □

It is interesting to also note that the above constructions allow us to use any switch set of the form $\{\frac{1}{2},\frac{1}{3},\frac{1}{5},\ldots,\frac{1}{p_{\max}}\}$ to realize any distribution with a denominator that can be factored into primes $2,3,5,\ldots,p_{\max}$.

## 6 Robustness of Probability Generation

The above algorithms looked at probability generation given a fixed switch set of distributions. However, in physical systems, it may be the case that the generation of randomness is error-prone. If we want to use a pswitch with distribution $(p_0,p_1)$, the physical pswitch may actually have distribution $(p_0+\epsilon,p_1-\epsilon)$. Loh, Zhou, and Bruck looked at generating two-state probabilities given pswitches with errors [8]. They found that any binary distribution generated according to their algorithm, regardless of size, had error bounded by $2\epsilon$.

We examine the same problem in the context of multivalued distribution generation and show that a generalized result holds for any number of states. Define the error of a multivalued distribution as the largest error over all the states. That is, if a pswitch with desired distribution $(p_0, p_1,\ldots,p_{N-1})$ has an actual distribution of $(p_0+\epsilon_0,p_1+\epsilon_1,\ldots,p_{N-1}+\epsilon_{N-1})$, then the error of the switch is $\max_i|\epsilon_i|$.

We will begin by demonstrating robustness for $N$-state binary distributions generated according to the algorithm in Section 4. This algorithm uses switches from the switch set $\mathbf{S}=\{\frac{1}{2}\}$ as well as the deterministic switches. For our analysis, we allow errors on the active states of pswitches. As a result, deterministic switches have no errors since the sum of the single active probability must equal 1. The $\frac{1}{2}$-pswitch has distribution $(\frac{1}{2}+\hat{\epsilon},0,\ldots,0,\frac{1}{2}-\hat{\epsilon})$, $|\hat{\epsilon}|\leq\epsilon$.

**Lemma 3 (Error Bound on Boundary States).** *Generate any distribution $(\ldots,0,\frac{x_i}{2^n},\ldots,\frac{x_k}{2^n},0,\ldots)$ according to the binary $N$-state algorithm where state $i$ is the smallest active state and state $k$ is the largest active state. If we allow at most $\epsilon$ error on the pswitches in the switch set, then the actual distribution generated will be $(\ldots,0,\frac{x_i}{2^n}+\delta_i,\ldots,\frac{x_k}{2^n}+\delta_k,0,\ldots)$ where $|\delta_i|\leq 2\epsilon$, $|\delta_k|\leq 2\epsilon$.*

**Proof.** In each step of the algorithm, a distribution $r=(\ldots,r_i,\ldots,r_k,\ldots)$ is made out of two pswitches $p=(\ldots,p_i,\ldots,p_j,\ldots)$ and $q=(\ldots,q_j,\ldots,q_k,\ldots)$ where $i\leq j\leq k$. The composition is in the form $r=p+(\frac{1}{2},0,\ldots,0,\frac{1}{2})q$. We will prove robustness using induction on switches $p$ and $q$.

Base case. For any deterministic distribution, the lemma is trivially satisfied.

Inductive step. Assume we are given $p$ and $q$ satisfying the inductive hypothesis. That is, for $p=(\ldots,p_i+\Delta_i,\ldots,p_j+\Delta_j,\ldots)$, $q=(\ldots,q_j+\delta_j,\ldots,q_k+\delta_k,\ldots)$, $|\Delta_i|$, $|\Delta_j|,|\delta_j|,|\delta_k|\leq 2\epsilon$. Then the errors for states $i$ and $k$ on distribution $r=p+(\frac{1}{2}+\hat{\epsilon},0,\ldots,0,\frac{1}{2}-\hat{\epsilon})q$ can be calculated as follows:

$$
\begin{aligned}
|\text{error}_{i<j}| &= \left|\left(\frac{1}{2}+\hat{\epsilon}\right)(p_i+\Delta_i)-\frac{1}{2}p_i\right| \\
&= \left|\frac{1}{2}\Delta_i+\hat{\epsilon}(p_i+\Delta_i)\right| \\
&\leq \frac{1}{2}|\Delta_i|+|\hat{\epsilon}||p_i+\Delta_i|\leq 2\epsilon \\
|\text{error}_{i=j}| &= \left|\left(\frac{1}{2}+\hat{\epsilon}\right)+\left(\frac{1}{2}-\hat{\epsilon}\right)(q_i+\delta_i)-\left(\frac{1}{2}+\frac{1}{2}q_i\right)\right| \\
&= \left|\frac{1}{2}\delta_i+\hat{\epsilon}(1-(q_i+\delta_i))\right| \\
&\leq \frac{1}{2}|\delta_i|+|\hat{\epsilon}||1-(q_i+\delta_i)|\leq 2\epsilon \\
|\text{error}_{k>j}| &= \left|\left(\frac{1}{2}-\hat{\epsilon}\right)(q_k+\delta_k)-\frac{1}{2}q_k\right| \\
&= \left|\frac{1}{2}\delta_k-\hat{\epsilon}(q_k+\delta_k)\right| \\
&\leq \frac{1}{2}|\delta_k|+|\hat{\epsilon}||q_k+\delta_k|\leq 2\epsilon \\
|\text{error}_{k=j}| &= \left|\left(\frac{1}{2}-\hat{\epsilon}\right)+\left(\frac{1}{2}+\hat{\epsilon}\right)(p_k+\Delta_k)-\left(\frac{1}{2}+\frac{1}{2}p_k\right)\right| \\
&= \left|\frac{1}{2}\Delta_k-\hat{\epsilon}(1-(p_k+\Delta_k))\right| \\
&\leq \frac{1}{2}|\Delta_k|+|\hat{\epsilon}||1-(p_k+\Delta_k)|\leq 2\epsilon.
\end{aligned}
$$

Then we find that the errors for states $i$ and $k$ still satisfy the inductive hypothesis, so we are done. □

Using this lemma, we are able to prove robustness of our algorithms for both binary and rational probability distributions. Note that the robustness of the state reduction algorithm for rational distributions can be shown to follow from the same proof as that of the binary robustness result. We leave the proofs to the appendix, available in the online supplemental material, because they are simply an inductive proof that uses algebraic manipulations similar to those in the previous lemma.

**Theorem 6 (Robustness of Binary N-state Distributions).** *Generate any distribution $(\frac{x_0}{2^n},\frac{x_1}{2^n},\ldots,\frac{x_{N-1}}{2^n})$ according to the binary $N$-state algorithm. If we allow the pswitches in the*

| a | | b | | c |
|---|---|---|---|---|
| $r_0.r_3\, r_2\, r_1$ | $U_3$ | $r_0.r_2\, r_1$ | $U_2$ | B-algorithm |
| 0 . 0 0 0 | (0/8, 8/8) | 0 . 0 0 | (0/4, 4/4) | (0/4, 4/4) + (1/2, 1/2)(0,1) |
| 0 . 0 0 1 | (1/8, 7/8) | 0 . 0 1 | (1/4, 3/4) | (1/4, 3/4) + (1/2, 1/2)(0,1) |
| 0 . 0 1 0 | (2/8, 6/8) | 0 . 1 0 | (2/4, 2/4) | (2/4, 2/4) + (1/2, 1/2)(0,1) |
| 0 . 0 1 1 | (3/8, 5/8) | 0 . 1 1 | (3/4, 1/4) | (3/4, 1/4) + (1/2, 1/2)(0,1) |
| 0 . 1 0 0 | (4/8, 4/8) | 0 . 0 0 | (0/4, 4/4) | (1, 0) + (1/2, 1/2)(0/4,4/4) |
| 0 . 1 0 1 | (5/8, 3/8) | 0 . 0 1 | (1/4, 3/4) | (1, 0) + (1/2, 1/2)(1/4,3/4) |
| 0 . 1 1 0 | (6/8, 2/8) | 0 . 1 0 | (2/4, 2/4) | (1, 0) + (1/2, 1/2)(2/4,2/4) |
| 0 . 1 1 1 | (7/8, 1/8) | 0 . 1 1 | (3/4, 1/4) | (1, 0) + (1/2, 1/2)(3/4,1/4) |
| 1 . 0 0 0 | (8/8, 0/8) | 1 . 0 0 | (4/4, 0/4) | (1, 0) + (1/2, 1/2)(4/4,0/4) |

Fig. 10. *Two-state universal probability generator (UPG) mappings.* (a) A mapping for a UPG that generates distributions of the form $(\frac{x_0}{8}, \frac{x_1}{8})$. (b) Removing $r_3$, we look at the inputs to a UPG that generates distributions of the form $(\frac{x_0}{4}, \frac{x_1}{4})$. (c) Notice that the B-algorithm decompositions of the probabilities of part (a) correspond to those in part (b).

*switch set* $\mathbf{S} = \frac{1}{2}$ *to have up to* $\epsilon$ *error in the active states, then the actual distribution* $(\frac{x_0}{2^n} + \delta_0, \frac{x_1}{2^n} + \delta_1, \ldots, \frac{x_{N-1}}{2^n} + \delta_{N-1})$ *has errors* $|\delta_i| \leq 3\epsilon$, $|\delta_0|, |\delta_{N-1}| \leq 2\epsilon$.

**Theorem 7 (Robustness of Rational Distributions).** *Generate any distribution* $(\frac{x_0}{q^n}, \frac{x_1}{q^n}, \ldots, \frac{x_{N-1}}{q^n})$ *according to the denominator reduction algorithm. If we allow the pswitches in the switch set* $\mathbf{S} = \{\frac{1}{2}, \ldots, \frac{1}{q}\}$ *to have up to* $\epsilon$ *error in the active states (e.g.* $|\hat{\epsilon}| \leq \epsilon$*), then the actual distribution* $(\frac{x_0}{q^n} + \delta_0, \frac{x_1}{q^n} + \delta_1, \ldots, \frac{x_{N-1}}{q^n} + \delta_{N-1})$ *has errors* $|\delta_i| \leq (q+1)\epsilon$, $|\delta_0|$, $|\delta_{N-1}| \leq q\epsilon$.

## 7 Universal Probability Generator

Up till now, we have only looked at circuits with set switches and described algorithms for realizing specific probability distributions. A next question is: What about circuits that can implement various distributions based on input switches? Wilhelm and Bruck approached this problem for the two-state version [21] by constructing a circuit which they called a Universal Probability Generator (UPG). This circuit maps $n$ deterministic bits into output probabilities of the form $\frac{x_0}{2^n}$ in increasing order. In other words, *any* probability with denominator $2^n$ can be generated by setting the input bits to its binary representation.

This functionality alone is not surprising since it can easily be done with an exponential number of switches using a tree-like structure: take the circuit for each probability and hook them up together with the appropriate input switches. The remarkable result is that the UPG only requires a linear number of switches in $n$.

In this section we generalize their result to creating a $N$-state UPG. The $N$-state UPG will be able to realize any binary probability distribution of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$ using $O(n^{N-1})$ switches.

### 7.1 An Alternate Proof for the Two-State UPG

The discovery of the two-state UPG found in [21] is through a stroke of genius: A circuit is proposed, and then an inductive argument is applied to prove the correctness of the circuit. Here, we will provide a new methodical derivation of their two-state UPG, which will give us the intuition required to generalize to $N$ states. Our strategy is to first construct a naive circuit that uses an exponential number of switches, but correctly implements our desired functionality. Then we will use algebraic rules to reduce the algebraic representation of the circuit to simpler forms, the last of which will coincide with the circuit presented by Wilhelm and Bruck.

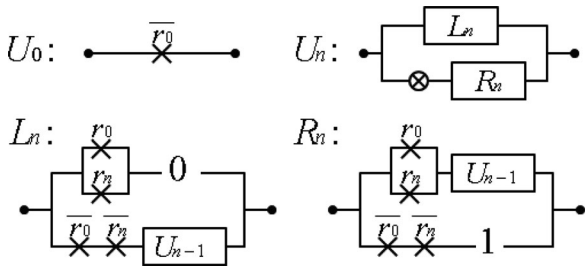



Fig. 11. *Two-state exponential UPG.* This is a UPG derived directly from the B-algorithm. It uses an exponential number of switches since $U_n$ uses two copies of $U_{n-1}$ in its recursive construction.

*Definition. (Two-state UPG).* A two-state UPG $U_n$ is a circuit that realizes distributions of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n})$ using $n+1$ input bits which we will refer to as $r_0$, $r_1$, ..., $r_n$. $r_0$ will represent the deterministic bit, while $r_n, r_{n-1}, \ldots, r_2, r_1$ will represent the fractional bits. By appropriately setting $r_i$, the circuit $U_n$ will realize the corresponding distribution.

As an example, we look at the input-output mappings for the circuit $U_3$. If we input $\mathbf{r} = 0001$ (i.e., $r_0 = 0$, $r_3 = 0, r_2 = 0, r_1 = 1$), the circuit will realize $(\frac{1}{8}, \frac{7}{8})$ since $\frac{1}{8} = 0.001_2$. The input 0101 will realize $(\frac{5}{8}, \frac{3}{8})$ since $\frac{5}{8} = 0.101_2$ (see Fig. 10a).

The motivation for the UPG circuit comes from an interesting property in the truth table. For example, let us enumerate all the outputs for $U_3$ (Fig. 10a). For each row (input), we ask the following questions: What would the output of $U_2$ be given the inputs $r_0, r_2, r_1$ that were used for $U_3$? Is there a relationship to the construction of the $U_3$ output probability?

If we calculate these outputs, we find that they are the same probability distributions used in the binary algorithm for two-states (Figs. 10b and 10c). From here, the (exponential) recursive construction is straightforward.

**Lemma 4.** *A two-state UPG* $U_n$ *with inputs* $r_0, r_n, \ldots, r_2, r_1$ *can be constructed with an exponential number of switches using the recursive construction in Fig. 11, where the* $n$ *bits used in* $U_{n-1}$ *are* $r_0, r_{n-1}, \ldots, r_2, r_1$. *Specifically,*

$$U_n = L_n + \left(\frac{1}{2}, \frac{1}{2}\right) R_n,$$
$$L_n = r_{0n} \cdot 0 + \bar{r}_{0n} U_{n-1},$$
$$R_n = r_{0n} U_{n-1} + \bar{r}_{0n} \cdot 1,$$

*where* $r_{0n} = r_0 + r_n$ *and* $\bar{r}_{0n} = \bar{r}_0 \bar{r}_n$.

**Proof.** This can be seen from Fig. 10, but an inductive proof is detailed in the appendix, available in the online supplemental material. □

**Lemma 5.** *A two-state UPG* $U_n$ *with inputs* $r_0, r_n, \ldots, r_2, r_1$ *can be constructed with a linear number of switches using either of the two recursive constructions in Fig. 12, where the* $n$ *bits used in* $U_{n-1}$ *are* $r_0, r_{n-1}, \ldots, r_2, r_1$.

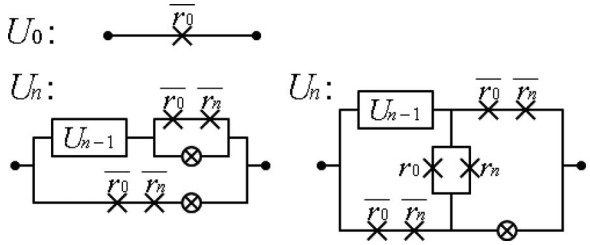



Fig. 12. *Two-state UPG reduced.* After reducing the exponential UPG, we get two linear UPGs. One is an sp-circuit and uses two stochastic switches. The other is non-sp and uses one stochastic switch.

**Proof.** We will prove this by algebraically reducing the exponential UPG. Let $p_i = (\frac{1}{2}, \frac{1}{2})$ be i.i.d. Then,

$$\begin{aligned} U_n &= L_n + p_n R_n \\ &= r_{0n} \cdot 0 + \bar{r}_{0n} U_{n-1} + p_n(r_{0n} U_{n-1} + \bar{r}_{0n} \cdot 1) \\ &= \bar{r}_{0n} U_{n-1} + p_n(r_{0n} U_{n-1} + \bar{r}_{0n}) \\ &= U_{n-1}(\bar{r}_{0n} + r_{0n} p_n) + \bar{r}_{0n} p_n \\ &= U_{n-1}(\bar{r}_{0n} + p_n) + \bar{r}_{0n} p_n. \end{aligned}$$

This is exactly the form of the series parallel construction in Fig. 12. One caveat to note is that we have two copies of $p_n$ in our proof, which would suggest that they need to be jointly distributed. It turns out that we can actually use two independent switches as shown in Fig. 12. The reason is because switches $\hat{r}_0$ and $\hat{r}_n$ ensure that only one of the switches is ever used.

$$\begin{aligned} U_n &= L_n + p_n R_n \\ &= r_{0n} \cdot 0 + \bar{r}_{0n} U_{n-1} + p_n(r_{0n} U_{n-1} + \bar{r}_{0n} \cdot 1) \\ &= \bar{r}_{0n} U_{n-1} + p_n(r_{0n} U_{n-1} + \bar{r}_{0n}) \\ &= [U_{n-1}][\bar{r}_{0n}] + [\bar{r}_{0n}][r_{0n}][\bar{r}_{0n}] \\ &\quad + [U_{n-1}][r_{0n}][p_n] + [\bar{r}_{0n}][p_n]. \end{aligned}$$

This is exactly the form of the non-series parallel construction in Fig. 12 □

The result thus far is nice, but one feels somewhat unsatisfied at the number of times $r_0$ must be used. We solve this problem to get to the final form of Wilhelm and Bruck's two-state UPG.

**Theorem 8.** *A two-state UPG $U_n$ with inputs $r_0, r_n, \ldots, r_2, r_1$ can be constructed with a linear number of switches using either of the two recursive constructions in Fig. 14, where the n bits used in $U_{n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$.*

**Proof.** We prove that the circuits in Figs. 12 and 14 are equivalent by using induction. Base case. $U_0 = \bar{r}_0 = \bar{r}_0 U'_0$.

Inductive step. Assume that $U_{n-1} = r_0 U'_{n-1}$. Then,

$$\begin{aligned} U_n &= U_{n-1}(\bar{r}_n \bar{r}_0 + p_n) + p_n \bar{r}_n \bar{r}_0 \\ &= \bar{r}_0 U'_{n-1}(\bar{r}_n \bar{r}_0 + p_n) + p_n \bar{r}_n \bar{r}_0 \\ &= \bar{r}_0(U'_{n-1}(\bar{r}_n \bar{r}_0 + p_n) + p_n \bar{r}_n) \\ &= \bar{r}_0(U'_{n-1}(\bar{r}_n + p_n) + p_n \bar{r}_n) = \bar{r}_0 U'_n. \end{aligned}$$

□

In this final form, the series-parallel circuit uses $2n$ pswitches and $2n+1$ deterministic switches. The non-sp construction uses $n$ pswitches and $3n+1$ deterministic switches.

Before we can generalize to $N$ states, we need to define some new notation and equalities. Define the two-state UPG $U_{[i,i+1],n}$ to be a circuit generating probability $\frac{x_0}{2^n}$ on state $i$ and $\frac{x_1}{2^n}$ on state $i+1$.

Under this notation, the previous results we proved were for $U_{[0,1],n}$. We can extend these results to $U_{[i,i+1],n}$ with the following changes:

1) The input bits $\mathbf{r}$ will take on values $i$ and $i+1$ instead of 0 and 1 respectively
2) The pswitch $p_n$ will take on values $i$ and $i+1$ with $\frac{1}{2}$ probability each instead of taking on values 0, 1 with equal probability.

But what if we didn't use values $i$ and $i+1$ for $\mathbf{r}, p_n$?

**Lemma 6.** *Let $U_{[i,i+1],n}(a,b)$ be a circuit identical to $U_{[i,i+1],n}$ except that $\mathbf{r}, p_n$ take on values $a$ and $b$. Our previous results for probability generation are for $U_{[i,i+1],n}(i, i+1)$. Then if $a \leq i$, $b \geq i+1$,*

$$U_{[i,i+1],n}(i, i+1) = (i + U_{[i,i+1],n}(a,b))(i+1).$$

**Proof.** $U_{[i,i+1],n}(a,b)$ will realize the desired distribution on states $a$ and $b$. Then if $a \leq i$, $b \geq i+1$, it is trivially that taking the max of $i$ and the min of $i+1$ will give us the same distribution on states $i$ and $i+1$. □

We want to avoid this messy notation for future generalizations. All future instances of $U_{[i,i+1],n}$ are actually representing $(i + U_{[i,i+1],n})(i+1)$. In other words, $U_{[i,i+1],n}$ will always generate the 'correct' distribution on states $i$ and $i+1$ as long as $\mathbf{r}, p_n$ take on values $a \leq i$ and $b \geq i+1$.

### 7.2 Three-State UPG

The three-state UPG can now be derived in the same way. Because the proofs are almost identical in structure, we will only detail them in the appendix, available in the online supplemental material. In essence, we first construct an exponential three-state UPG that follows closely from the algorithm for realizing three-state distributions. Then, we will algebraically reduce it to a quadratic construction and remove repetitive bits.

*Definition. (Three-state UPG).* A Three-state UPG $U_{[0,2],n}$ is a circuit that generates distributions of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \frac{x_2}{2^n})$ using two sets of $n+1$ input bits $\mathbf{r}, \mathbf{s}$. The notation $[0,2]$ in the subscript makes it explicit that the states we are generating a distribution for are 0, 1, and 2. The input bits $\mathbf{r}$ and $\mathbf{s}$ will take on values 0 or 2; when the bits $\mathbf{r}$, in the order $r_0, r_n, r_{n-1}, \ldots, r_2, r_1$ are set to the binary representation of $\frac{x_0}{2^n}$ and the bits $\mathbf{s}$, in the order $s_0, s_n, s_{n-1}, \ldots, s_2, s_1$ are set to the binary representation of $\frac{x_0}{2^n} + \frac{x_1}{2^n}$ (with the symbol 2 replacing the Boolean 1), then the circuit $U_{[0,2],n}$ will realize distribution $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \frac{x_2}{2^n})$. In other words, to realize any desired binary probability $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \frac{x_2}{2^n})$, we set the input bits $\mathbf{r}$ and $\mathbf{s}$ to $p_0$ and $p_0 + p_1$ respectively.

As an example, we look at the input-output mappings for the circuit $U_{[0,2],2}$. If we input $\mathbf{r} = 002, \mathbf{s} = 020$, the circuit will realize $(\frac{1}{4}, \frac{1}{4}, \frac{1}{2})$ since $\frac{1}{4} = 0.01_2$ and $\frac{1}{4} + \frac{1}{4} = \frac{1}{2} = 0.10_2$. The

| a | | | b | | | | | c |
|---|---|---|---|---|---|---|---|---|
| $r_0.r_2\,r_1$ | $s_0.s_2\,s_1$ | $U_{[0,2],2}$ | $r_0.r_1$ | $s_0.s_1$ | $U_{[0,1],1}$ | $U_{[1,2],1}$ | $U_{[0,2],1}$ | Binary 3-state alg. |
| 0 . 0 0 | 0 . 0 0 | (0/4, 0/4, 4/4) | 0 . 0 | 0 . 0 | | | (0/2, 0/2, 2/2) | (0/2, 0/2, 2/2) + 1/2(0, 0, 1) |
| 0 . 0 0 | 0 . 0 2 | (0/4, 1/4, 3/4) | 0 . 0 | 0 . 2 | | | (0/2, 1/2, 1/2) | (0/2, 1/2, 1/2) + 1/2(0, 0, 1) |
| 0 . 0 0 | 0 . 2 0 | (0/4, 2/4, 2/4) | 0 . 0 | 0 . 0 | (0/2, 2/2, 0) | (0, 0/2, 2/2) | | (0/2, 2/2, 0) + 1/2 (0, 0/2, 2/2) |
| 0 . 0 0 | 0 . 2 2 | (0/4, 3/4, 1/4) | 0 . 0 | 0 . 2 | (0/2, 2/2, 0) | (0, 1/2, 1/2) | | (0/2, 2/2, 0) + 1/2 (0, 1/2, 1/2) |
| 0 . 0 0 | 2 . 0 0 | (0/4, 4/4, 0/4) | 0 . 0 | 2 . 0 | (0/2, 2/2, 0) | (0, 2/2, 0/2) | | (0/2, 2/2, 0) + 1/2 (0, 2/2, 0/2) |
| 0 . 0 2 | 0 . 0 2 | (1/4, 0/4, 3/4) | 0 . 2 | 0 . 2 | | | (1/2, 0/2, 1/2) | (1/2, 0/2, 1/2) + 1/2(0, 0, 1) |
| 0 . 0 2 | 0 . 2 0 | (1/4, 1/4, 2/4) | 0 . 2 | 0 . 0 | (1/2, 1/2, 0) | (0, 0/2, 2/2) | | (1/2, 1/2, 0) + 1/2 (0, 0/2, 2/2) |
| 0 . 0 2 | 0 . 2 2 | (1/4, 2/4, 1,4) | 0 . 2 | 0 . 2 | (1/2, 1/2, 0) | (0, 1/2, 1/2) | | (1/2, 1/2, 0) + 1/2 (0, 1/2, 1/2) |
| 0 . 0 2 | 2 . 0 0 | (1/4, 3/4, 0/4) | 0 . 2 | 2 . 0 | (1/2, 1/2, 0) | (0, 2/2, 0/2) | | (1/2, 1/2, 0) + 1/2 (0, 2/2, 0/2) |
| 0 . 2 0 | 0 . 2 0 | (2/4, 0/4, 2/4) | 0 . 0 | 0 . 0 | | | (0/2, 0/2, 2/2) | (1, 0, 0) + 1/2(0/2, 0/2, 2/2) |
| 0 . 2 0 | 0 . 2 2 | (2/4, 1/4, 1/4) | 0 . 0 | 0 . 2 | | | (0/2, 1/2, 1/2) | (1, 0, 0) + 1/2(0/2, 1/2, 1/2) |
| 0 . 2 0 | 2 . 0 0 | (2/4, 2/4, 0/4) | 0 . 0 | 2 . 0 | | | (0/2, 2/2, 0/2) | (1, 0, 0) + 1/2(0/2, 2/2, 0/2) |
| 0 . 2 2 | 0 . 2 2 | (3/4, 0/4, 1/4) | 0 . 2 | 0 . 2 | | | (1/2, 0/2, 1/2) | (1, 0, 0) + 1/2(1/2, 0/2, 1/2) |
| 0 . 2 2 | 2 . 0 0 | (3/4, 1/4, 0/4) | 0 . 2 | 2 . 0 | | | (1/2, 1/2, 0/2) | (1, 0, 0) + 1/2(1/2, 1/2, 0/2) |
| 2 . 0 0 | 2 . 0 0 | (4/4, 0/4, 0/4) | 2 . 0 | 2 . 0 | | | (2/2, 0/2, 0/2) | (1, 0, 0) + 1/2(2/2, 0/2, 0/2) |

Fig. 13. *Three-state UPG mapping.* The mapping for a UPG that generates distributions of the form $(\frac{x_0}{4}, \frac{x_1}{4}, \frac{x_2}{4})$. Here we notice that $y_0$ is being used a couple of times. Again, we don't make predictions about inputs that don't correspond to a probability distribution: this includes both those with a sum greater than 1 and those for which the $y$ encoding is less than the $x$ encoding (which would mean a negative $\frac{x_1}{2^n}$ value).

input $\mathbf{r} = 020, \mathbf{s} = 022$ will realize $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ since $\frac{1}{2} = 0.10_2$ and $\frac{1}{2} + \frac{1}{4} = \frac{3}{4} = 0.11_2$. (see Fig. 13a).

Again, the motivation for the exponential UPG comes from an interesting property in the truth table of $U_{[0,2],2}$. We first enumerate all outputs given inputs corresponding to valid probability distributions. For each row (input), we ask: What are the outputs of $U_{[0,1],1}, U_{[1,2],1}, U_{[0,2],1}$ if we use the inputs $r_0, r_1$ for $U_{[0,1],1}$, the inputs $s_0, s_1$ for $U_{[1,2],1}$, and $r_0$, $r_1, s_0, s_1$ for $U_{[0,2],2}$? Is there a relationship to the construction of the $U_{[0,2],1}$ output probability? The answer is yes. We find that they are the same probability distributions that are used in the binary algorithm for three-states (Figs. 13b and 13c).

**Lemma 7.** *A three-state UPG $U_{[0,2],n}$ with inputs $r_0, r_n, \ldots, r_2, r_1$ and $s_0, s_n, \ldots, s_2, s_1$ can be constructed with an exponential number of switches using the recursive construction in Fig. 15, where the bits used in $U_{[0,1],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$, the bits used in $U_{[1,2],n-1}$ are $s_0, s_{n-1}, \ldots, s_2, s_1$, and the bits used in $U_{[0,2],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$ and $s_0, s_{n-1}, \ldots, s_2, s_1$.*

Now that we have a three-state exponential UPG, we can use algebraic manipulation like what was used in the two-state case to reduce the exponential number of switches to our final form.

**Lemma 8.** *A three-state UPG $U_{[0,2],n}$ with inputs $r_0, r_n, \ldots, r_2, r_1$ and $s_0, s_n, \ldots, s_2, s_1$ can be constructed with a quadratic number of switches using either of the two recursive constructions in Fig. 16, where the bits used in $U_{[0,1],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$, the bits used in $U_{[1,2],n-1}$ are $s_0, s_{n-1}, \ldots, s_2, s_1$, and the bits used in $U_{[0,2],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$ and $s_0, s_{n-1}, \ldots, s_2, s_1$.*

**Theorem 9.** *A three-state UPG $U_{[0,2],n}$ with inputs $r_0, r_n, \ldots, r_2, r_1$ and $s_0, s_n, \ldots, s_2, s_1$ can be constructed with a quadratic number of switches using either of the two recursive constructions in Fig. 17, where the bits used in $U_{[0,1],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$, the bits used in $U_{[1,2],n-1}$ are $s_0, s_{n-1}, \ldots, s_2, s_1$, and the bits used in $U_{[0,2],n-1}$ are $r_0, r_{n-1}, \ldots, r_2, r_1$ and $s_0, s_{n-1}, \ldots, s_2, s_1$.*

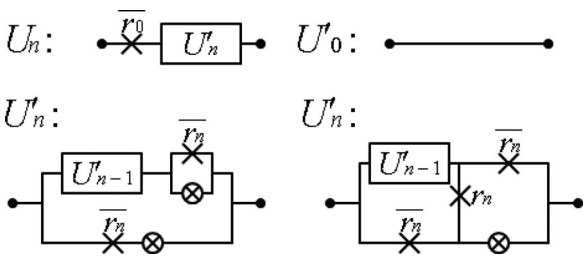



Fig. 14. *Two-state UPG removed bit.* Even though $r_0$ was important in controlling the cases of our algorithm, it turns out we can remove it from the recursive construction and just append it at the end. Note that $U'_n$ only uses bits $r_n, \ldots, r_2, r_1$.

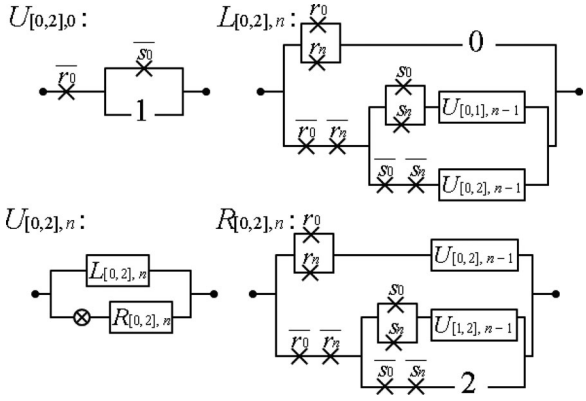



Fig. 15. *Three-state exponential UPG.* We design a UPG directly from the algorithm. Note that this uses an exponential number of switches since the recursion uses two copies of $U_{[0,2],n-1}$.

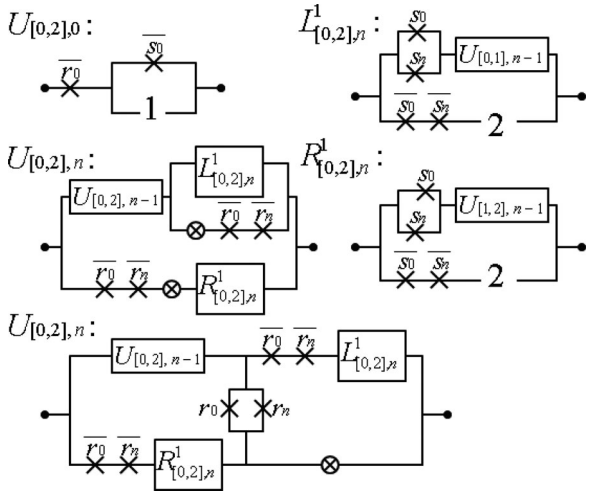


Fig. 16. *Three-state UPG reduced.* After reducing the exponential UPG, we get two linear UPGs. One is an sp-circuit and uses two stochastic switches. The other is non-sp and uses one stochastic switch.

We will use the notation $U_{[i,i+2],n}$ to denote a three-state UPG over the states $i, i+1, i+2$.

### 7.3 $N$-State UPG

The steps are proofs for generalizing to $N$ states are identical to the three-state version except that the variables $L$ and $R$ are generalized. Therefore, we will simply state the theorem and circuit generalization.

*Definition. ($N$-state UPG).* An $N$-state UPG $U_{[0,N-1],n}$ is a circuit that generates distributions of the form $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$ using $N-1$ input vectors $\mathbf{r_0}, \mathbf{r_1}, \ldots, \mathbf{r_{N-2}}$, where each vector $\mathbf{r_i}$ has $n+1$ bits $r_{i0}, r_{i1}, \ldots, r_{in}$. When the input vectors $\mathbf{r_i} = (r_{i0}, r_{in}, r_{i(n-1)}, \ldots, r_{i2}, r_{i1})$ are set to the binary representation of $\frac{x_0+x_1+\cdots+x_i}{2^n}$ with the symbol $N-1$ replacing the Boolean 1, then the circuit $U_{[0,N-1],n}$ will realize distribution $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$. In other words, to generate any desired binary distribution $(\frac{x_0}{2^n}, \frac{x_1}{2^n}, \ldots, \frac{x_{N-1}}{2^n})$, we set the input vector $\mathbf{r_0}$ to $\frac{x_0}{2^n}$, the input vector $\mathbf{r_1}$ to the sum $\frac{x_0+x_1}{2^n}, \ldots,$ and the input vector $\mathbf{r_{N-2}}$ to the sum $\frac{x_0+x_1+\ldots+x_{N-2}}{2^n}$.

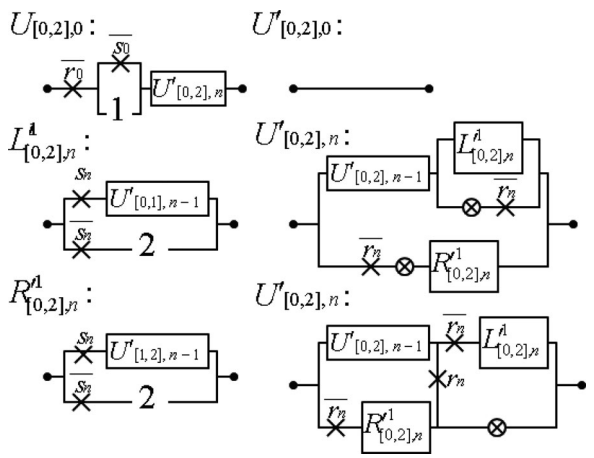


Fig. 17. *Three-state UPG removed bit.* Even though $r_0$ and $s_0$ were important in controlling the cases of our algorithm, it turns out we can remove it from the recursive construction and just append it at the end.

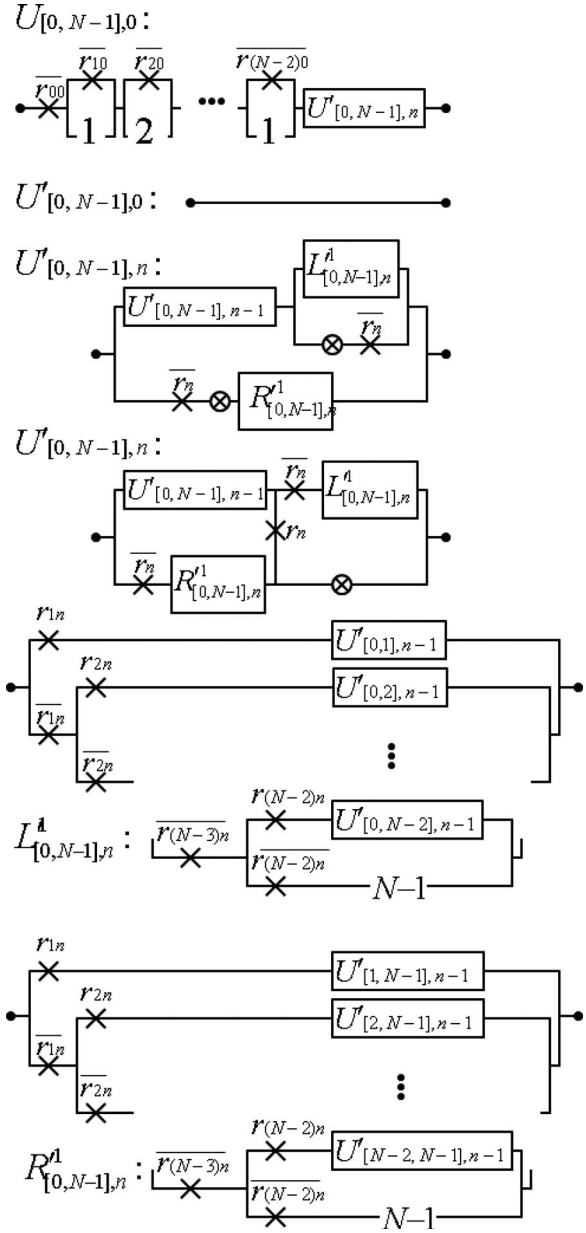


Fig. 18. *$N$-state UPG.* The $N$-state UPG is almost exactly the same as the three-state UPG.

**Theorem 10.** *An $N$-state UPG $U_{[0,N-1],n}$ with inputs $\mathbf{r_i}, 0 \le i \le N-2$ can be constructed with a polynomial number of switches $O(n^{N-1})$ using either of the two recursive constructions in Fig. 18, where the bits used in $U_{[0,i],n-1}$ are $\mathbf{r_0}, \mathbf{r_1}, \ldots, \mathbf{r_{i-1}}$ and the bits used in $U_{[i,N-1],n-1}$ are $\mathbf{r_i}, \mathbf{r_{i+1}}, \ldots, \mathbf{r_{N-2}}$.*

## 8 Partial Orders

It is interesting to consider extending this work to states in a partial order. Partial orders are orderings where two states may not be comparable with the "greater than or equal to" relation. For example, in the partial order of Fig. 19b, $x_{11}$ is greater than all other states and $x_{00}$ is less than all other states, but $x_{01}$ and $x_{10}$ are incomparable.

For partial orders, one natural generalization of max and min are the $\vee$ (join) and $\wedge$ (meet) operators. The $\vee$ of states $a$ and $b$ is the least upper bound on $a$ and $b$ while the $\wedge$ of states $a$ and $b$ is the greatest lower bound on $a$ and $b$. When

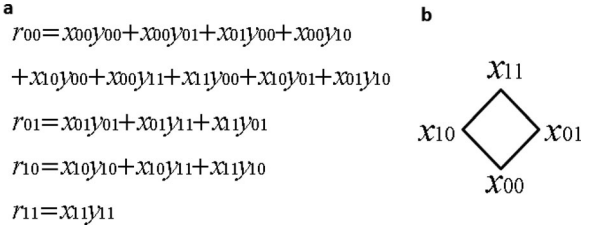



Fig. 19. *A simple partially ordered lattice.* This is the simplest partial order, so results extend to all others. (a) An example of a series composition (meet) of switches $x$ and $y$. (b) The lattice structure.

parallel and series connections are generalized from max and min to join and meet respectively, we find that we cannot generate many distributions using series-parallel circuits. We demonstrate this for the partial order represented by Fig. 19b, but the result extends to any partial order.

**Theorem 11 (Partial Order Inexpressibility).** *For the lattice in Fig. 19b, no distributions of the form $v = (0, 1-p, p, 0)$, where $0 < p < 1$, are realizable by building an sp circuit with any switch set unless $(0, 1-p, p, 0)$ itself is in the switch set.*

**Proof.** Assume that $v$ is realizable by a series composition. That is, $v = xy$. Then we have the following expressions for $v$ in terms of $x$ and $y$:

$$v_{00} = x_{00} + y_{00} - x_{00}y_{00} + x_{01}y_{10} + x_{10}y_{01} \tag{3}$$

$$v_{10} = x_{10}y_{11} + x_{11}y_{10} + x_{10}y_{10} \tag{4}$$

$$v_{01} = x_{01}y_{11} + x_{11}y_{01} + x_{01}y_{01} \tag{5}$$

$$v_{11} = x_{11}y_{11}. \tag{6}$$

Since $v_{00} = 0$, we can conclude from (3) that we must have $x_{00} + y_{00} - x_{00}y_{00} = 0$, which implies that $x_{00} = 0$ and $y_{00} = 0$. Similarly, since $v_{11} = 0$, we can conclude from (6) that we must have $x_{11} = 0$ or $y_{11} = 0$.

Without loss of generality, suppose $x_{11} = 0$. Plugging into (4) and (5), we get

$$v_{10} = x_{10}y_{11} + x_{10}y_{10} = x_{10}(y_{11} + y_{10})$$
$$v_{01} = x_{01}y_{11} + x_{01}y_{01} = x_{01}(y_{11} + y_{01}).$$

But since $v_{10} = p > 0$ and $v_{01} = 1 - p > 0$, we know that $x_{10} \neq 0$ and $x_{01} \neq 0$.

Finally, recall that $x_{11} = 0$ without loss of generality and that $v_{00} = 0$. Then $v_{00} = 0 = x_{01}y_{10} + x_{10}y_{01}$. Since both $x_{10}$ and $x_{01}$ are not equal to zero, this must mean that $y_{10} = 0$ and $y_{01} = 0$. Now, recall that we have already concluded that $y_{00} = 0$. Then we now know that $y_{11} = 1$. With this, we can simply plug the appropriate values for $y$ into equations (3-6) to conclude that $x = v$, which means that we can only realize $v$ through a series composition if we already have access to $v$. A similar argument will show the same result for a parallel composition. □

## 9 Applications

As alluded to in Section 1.5, the max-min multivalued alphabet is naturally suited for reasoning about timings and dependencies. Switches represent events, and the states represent the discrete time when the event occurs. When two switches $x$ and $y$ are composed to make the circuit $z$, we are saying that for event $z$ to occur, it depends on events $x$ and $y$. If event $z$ requires both $x$ and $y$ to occur before it can occur, then this can be represented by a parallel connection since the time $z$ occurs will be the $\max(x, y)$. If event $z$ only needs either $x$ or $y$ to occur, then this can be represented by a series connection since the time $z$ occurs will be the $\min(x, y)$.

Such timing based computations also appear in neural circuits. A rough description of how neurons work is that incoming signals from previous neurons will release chemicals that change the voltage potential of the neuron. If this voltage potential exceeds a threshold level, then the neuron will fire its own signal. Consider a network of neurons where each neuron only has incoming signals from two other neurons. Then, consider a neuron $x$ with incoming neurons $y$ and $z$. Neuron $x$ will fire when the incoming signals exceeds a certain threshold potential. If the threshold of $x$ is low, then if it receives an incoming signal from either $y$ or $z$, it will exceed the threshold and neuron $x$ will fire. Then the time would be the minimum of $y$ and $z$ firing times. If the threshold is high, we can imagine that neuron $x$ will only fire if it receives signals from both incoming neurons. The time would be the maximum of $y$ and $z$ firing times.

## 10 Conclusion

In this paper, we studied probability generation in multivalued relay circuits. We proved a duality result for these circuits and derived algorithms for generating any rational probability distribution. We show that errors in the designed circuits remain bounded regardless of the circuit size. Finally, we constructed a universal probability generator for mapping deterministic inputs to any desired probability distributions and demonstrated an impossibility result for partial orders. An interesting direction is to extend this work to better understand neural coding.

## Acknowledgments

The authors would like to thank Dan Wilhelm, Hongchao Zhou, and Ho-lin Chen for helpful discussions, and the Caltech SURF program, the Molecular Programming Project funded by the NSF Expeditions in Computing Program under grant CCF-0832824, and Aerospace Corporation for funding. D.T. Lee is the corresponding author.

## References

[1] J. Abrahams, "Generation of discrete distributions from biased coins," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1541–1546 Sep. 1996.

[2] J. Bruck, "The logic of biological networks," Purdue Univ. (2006). http://video.google.com/videoplay?docid=3684954164278324696.

[3] M. Elowitz, A. Levine, E. Siggia, and P. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 297, pp. 183–186, 2002.

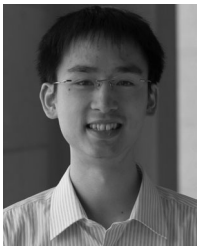
[4] J. Gunawardena, "Min-max functions," *Discrete Event Dyn. Syst.: Theory Appl.*, vol. 4, no. 4, pp. 377–407, 1994.

[5] A. Gill, "Synthesis of probability transformers," *J. Franklin Inst.*, vol. 274, no. 1, pp. 1–19, 1962.

[6] T. Han and M. Hoshi, "Interval algorithm for random number generation," *IEEE Trans. Inf. Theory*, vol. 43, no. 2, pp. 599–611, Mar. 1997.

[7] D. Knuth and A. Yao, "The complexity of nonuniform random number generation," *Algorithms Complexity: New Directions Recent Results*, pp. 357–428, 1976.

[8] P. Loh, H. Zhou, and J. Bruck, "The robustness of stochastic switching networks," in *Proc. IEEE Int. Symp. Inform. Theory*, 2009, pp. 2066–2070.
[9] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
[10] E. Post, "Introduction to a general theory of elementary propositions," *Am. J. Math.* vol. 43, no. 3, pp. 163–185, 1921.
[11] W. Qian, M. D. Riedel, H. Zhou, and J. Bruck, "Transforming probabilities with combinational logic," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* vol. 30, no. 9, pp. 1279–1292, Sep. 2011.
[12] D. Rine, *Computer Science and Multiple-Valued Logic: Theory and Applications*. Amsterdam, The Netherlands, North Holland, 1984.
[13] E. Schneidman, B. Freedman, and I. Segev, "Ion channel stochasticity may be critical in determining the reliability and precision of spike timing," *Neural Comput.* vol. 10, pp. 1679–1703, 1998.
[14] G. Shahaf, D. Eytan, A. Gal, E. Kermany, V. Lyakhov, C. Zrenner, and S. Marom, "Order-based representation in random networks of cortical neurons," *PLoS Comput. Biol.*, vol. 4, no. 11, p. e1000228, 2008.
[15] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Trans. Am. Inst. Elect. Eng.*, vol. 57, pp. 713–723, 1938.
[16] C. E. Shannon, "A mathematical theory of communication," *The Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.
[17] C. L. Sheng, "Threshold logic elements used as a probability transformer," *J. ACM*, vol. 12, no. 2, pp. 262–276, 1965.
[18] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with finite stochastic chemical reaction networks," *Nat. Comput.*, vol. 7, no. 4, pp. 615–633, 2008.
[19] J. von Neumann, "Various techniques used in connection with random digits," *Appl. Math. Ser.* vol. 12, pp. 36–38, 1951.
[20] D. Webb, "Many-valued logics," *Caltech Ph.D. dissertation*, Advisor Eric Bell, 1936.
[21] D. Wilhelm and J. Bruck, "Stochastic switching circuit synthesis," in *Proc. IEEE Int. Symp. Inform. Theory*, 2008, pp. 1388–1392.
[22] H. Zhou and J. Bruck, "On the expressibility of stochastic switching circuits," in *Proc. IEEE Int. Symp. Inform. Theory*, 2009, pp. 2061–2065.
[23] H. Zhou, H. Chen, and J. Bruck, "Synthesis of stochastic flow networks," *IEEE Trans. Comput.*, vol. 63, no. 5, pp. 1234–1247, May 2012.



**David T. Lee** received the BSc degree in electrical engineering from the California Institute of Technology, Pasadena, CA in 2010, and is currently working towards the PhD degree in electrical engineering at Stanford University, Stanford, CA. He is a recipient of the US National Science Foundation (NSF) Graduate Research Fellowship, a two-time recipient of the Brown Institute of Media Innovation Magic Grant, and an Accel Innovation Scholar. His current research interests include the application of randomness and computation to biological and social systems.

**Jehoshua Bruck** (S'86-M'89-SM'93-F'01) received the BSc and MSc degrees in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 1982 and 1985, respectively, and the PhD degree in electrical engineering from Stanford University, Stanford, CA, in 1989. He is the Gordon and Betty Moore professor of computation and neural systems and electrical engineering at the California Institute of Technology, Pasadena, CA. His extensive industrial experience includes working with IBM Almaden Research Center, as well as co-founding and serving as the chairman of Rainfinity, acquired by EMC in 2005; and XtremIO, acquired by EMC in 2012. His current research interests include information theory and systems and the theory of computation in biological networks. He is a recipient of the Feynman Prize for Excellence in Teaching, the Sloan Research Fellowship, the National Science Foundation Young Investigator Award, the IBM Outstanding Innovation Award, and the IBM Outstanding Technical Achievement Award. His papers were recognized in journals and conferences, including winning the 2010 IEEE Communications Society Best Student Paper Award in Signal Processing and Coding for Data Storage for his paper on codes for limited-magnitude errors in flash memories, the 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage for his paper on rank modulation for flash memories, the 2005 A. Schelkunoff Transactions Prize Paper Award from the IEEE Antennas and Propagation Society for his paper on signal propagation in wireless networks, and the 2003 Best Paper Award in the 2003 Design Automation Conference for his paper on cyclic combinational circuits. He is a fellow of the IEEE.