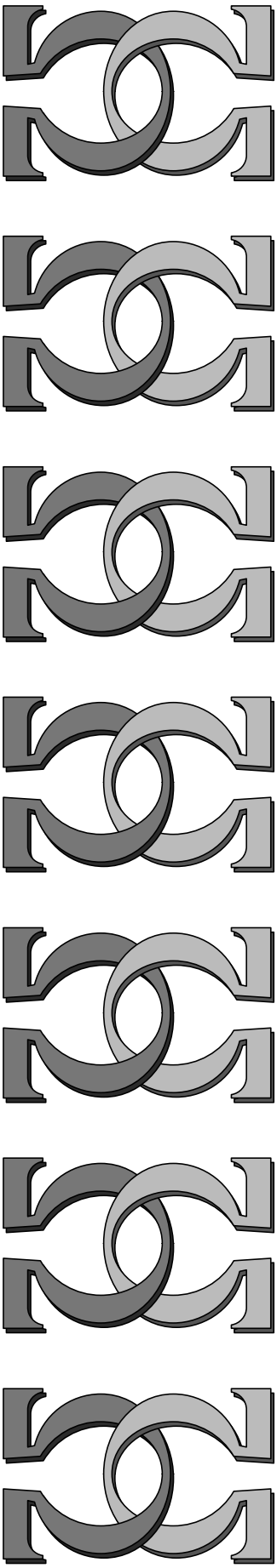

CDMTCS
Research Report Series

Testing Computational Complementarity for Mermin Automata

Cristian S. Calude
University of Auckland, New Zealand

Elena Calude
Massey University at Albany, New Zealand

Terry Chiou
University of Auckland, New Zealand

Monica Dumitrescu
Faculty of Mathematics, Bucharest University, Bucharest, Romania

Radu Nicolescu
University of Auckland, New Zealand

CDMTCS-109
July 1999

Centre for Discrete Mathematics and
Theoretical Computer Science

Testing Computational Complementarity for Mermin Automata

Cristian S. Calude,[*] **Elena Calude**,[†] **Terry Chiou**,[‡]
Monica Dumitrescu,[§] **Radu Nicolescu**[¶]

Abstract

Mermin [15] described a simple device to explain *Einstein-Podolsky-Rosen (EPR)* [12] correlations. This device was studied by means of a class of probabilistic (Mermin) automata in [4]. In [5] one shows that every deterministic automaton simulating with confidence 1/2 a probabilistic Mermin automaton features a classical behaviour. Is the above result true when the simulation is done at higher levels of confidence? To answer this question we study the distribution of two computational complementarity principles for two classes of deterministic automata which mimic the behaviour of Mermin's device with confidence in the intervals $(1/2, 11/16]$ and $(11/16, 7/8]$. Since the class of automata to be studied is large, it contains $9^{18} \approx 150 \cdot 10^{15}$ elements, we use simulation techniques. We show that, statistically, at any level of confidence $\alpha \in (1/2, 11/16]$, the class of deterministic automata simulating Mermin probabilistic automata display less correlations than typical deterministic automata with 9 states and 7 outputs, but at higher levels of confidence $\alpha \in (11/16, 7/8]$, when the simulation is more accurate, *deterministic automata simulating Mermin probabilistic automata display more correlations than typical deterministic automata with 9 states and 2 outputs.* In the last case, *EPR correlations established in* [4] *for Mermin probabilistic automata correspond to computational complementarity of the deterministic automata simulating Mermin probabilistic automata,* [10, 13, 18, 3, 6].

1 Introduction

Mermin probabilistic automata display strong correlations which model *Einstein-Podolsky-Rosen (EPR)* [12] correlations (see also [2, 17, 11, 1]) in the language of automata (see [4]). Is there any relation between these type of correlations and correlations modeled by computational complementarity in [10, 13, 18, 3, 6]? In what follows we study deterministic automata simulating, with various degrees of confidence, probabilistic automata modeling Mermin's device. In [5] one shows that every deterministic automaton simulating with confidence 1/2, a probabilistic Mermin automaton features a classical behaviour. Is the above result true when the simulation is done at higher levels of confidence? To answer this question we study the distribution of two computational complementarity principles for two classes of deterministic automata which simulate Mermin

[*]Institut für Theoretische Physik, University of Technology Vienna, Wiedner Hauptstraße 8-10/136, A-1040 Vienna, Austria; on leave from the Computer Science Department, The University of Auckland, Private Bag 92109, Auckland, New Zealand, e-mail: cristian@cs.auckland.ac.nz.

[†]Institute of Information Sciences, Massey University at Albany, Private Bag 102-904, North Shore MSC, Auckland, New Zealand, e-mail: E.Calude@massey.ac.nz.

[‡]Division of Science and Technology, Tamaki Campus, The University of Auckland, Private Bag 92109, Auckland, New Zealand, e-mail: terryc@ihug.co.nz.

[§]Department of Probability and Statistics, Faculty of Mathematics, Bucharest University, Str. Academiei 14, Bucharest, Romania, e-mail: mdumi@pro.math.unibuc.ro.

[¶]Division of Science and Technology, Tamaki Campus, The University of Auckland, Private Bag 92109, Auckland, New Zealand, e-mail: r.nicolescu@auckland.ac.nz.

probabilistic automata with confidence levels in the intervals $(1/2, 11/16]$ and $(11/16, 7/8]$. Since the class of automata to be studied is large—it contains $9^{18} \approx 150 \cdot 10^{15}$ elements—we use simulation techniques. The statistics obtained in this paper shows that at any level of confidence $\alpha \in (1/2, 11/16]$, the class of deterministic automata simulating Mermin probabilistic automata display less correlations than typical deterministic automata with 9 states and 7 outputs, but at higher levels of confidence $\alpha \in (11/16, 7/8]$, when the simulation is more accurate, *deterministic automata simulating Mermin probabilistic automata display more correlations than typical deterministic automata with 9 states and 2 outputs.* In the last case, *EPR correlations established in* [4] *for Mermin probabilistic automata correspond to computational complementarity of the deterministic automata simulating Mermin probabilistic automata,* [10, 13, 18, 3, 6].

2 Mermin's Device

Mermin [15] imagined a simple device to explain EPR conundrum without using the classical quantum mechanical notions of wave functions, superposition, wave-particle duality, uncertainty principle, etc. The device has three "completely unconnected", neither mechanical nor electromagnetic, parts, two detectors (D1) and (D2) and a source (S) emitting particles. The source is placed between the detectors: whenever a button is pushed on (S), shortly thereafter two particles emerge, moving off toward detectors (D1) and (D2). Each detector has a switch that can be set in one of three possible positions–labeled 1,2,3–and a bulb that can flash a red (R) or a green (G) light. The purpose of lights is to "communicate" information to the observer. Each detector flashes either red or green whenever a particle reaches it. Because of the lack of any relevant connections between any parts of the device, the link between the emission of particles by (S), i.e., as a result of pressing a button, and the subsequent flashing of detectors can only be provided by the passage of particles from (S) to (D1) and (D2). Additional tools can be used to check and confirm the lack of any communication, cf. [15], p. 941.

The device is repeatedly operated as follows:

1. the switch of either detector (D1) and (D2) is set randomly to 1 or 2 or 3, i.e., the settings or states 11, 12, 13, 21, 22, 23, 31, 32, 33 are equally likely,
2. pushing a button on (S) determines the emission toward both (D1) and (D2),
3. sometime later, (D1) and (D2) flash one of their lights, G or R,
4. every run is recorded in the form $ijXY$, meaning that (D1) was set to state i and flashed X and (D2) was set to j and flashed Y.

For example, the record $31GR$ means "(D1) was set to 3 and flashed G and (D2) was set to 1 and flashed R".

Long recorded runs show the following pattern:

a) For records starting with ii, i.e., 11, 22, 33, both (D1) and (D2) flash the same colours, RR, GG, with equal frequency; RG and GR are never flashed.

b) For records starting with $ij, i \neq j$, i.e., 12, 13, 21, 23, 31, 32, both (D1) and (D2) flash the same colour only $1/4$ of the time (RR and GG come with equal frequencies); the other $3/4$ of the time, they flash different colours (RG, GR), occurring again with equal frequencies.

The above patterns are statistical, that is they are subject to usual fluctuations expected in every statistical prediction: patterns are more and more "visible" as the number of runs becomes

larger and larger.

The conundrum posed by the existence of Mermin's device reveals as soon as we notice that the seemingly simplest physical explanation of the pattern a) is incompatible with pattern b). Indeed, as (D1) and (D2) are unconnected there is no way for one detector to "know", at any time, the state of the other detector or which colour the other is flashing. Consequently, it seems plausible to assume that the colour flashed by detectors is determined only by some property, or group of properties, of particles, say speed, size, shape, etc. What properties determine the colour does not really matter; only the fact that each particle carries a "program" which determines which colour a detector will flash in some state is important. So, we are led to the following two hypotheses:

H1 *Particles are classified into eight categories:*

$$GGG, GGR, GRG, GRR, RGG, RGR, RRG, RRR.^{1}$$

H2 *Two particles produced in a given run carry identical programs.*

According to H1–H2, if particles produced in a run are of type RGR, then both detectors will flash R in states 1 and 3; they will flash G if both are in state 2. Detectors flash the same colours when being in the same states because *particles carry the same programs.*

It is clear that from H1–H2 it follows that *programs carried by particles do not depend in any way on the specific states of detectors*: they are properties of particles not detectors. Consequently, both particles carry the same program whether or not detectors (D1) and (D2) are in the same states.[2]

A simple combinatorial argument shows that

[L] For each type of particle, *in runs of type* b) *both detectors flash the same colour at least one third of the time.*

The conundrum reveals as a significant difference appears between the data dictated by particle programs (colours agree at least one third of the time) and the quantum mechanical prediction (colours agree only one fourth of the time):

under H1–H2, *the observed pattern b) is incompatible with* [L].

3 Mermin's Probabilistic Automata

Mermin's device has been modeled by means of a probabilistic automaton $\mathcal{M}$ in [4, 5]. The states of the automaton are all combinations of states of detectors (D1) and (D2), $Q = \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$, the input alphabet models the lights, red and green, $\Sigma = \{G, R\}$, the output alphabet captures all combinations of lights flashed by (D1) and (D2), $O = \{GG, GR, RG, RR\}$, and the output function $f : Q \to O$, modeling all combinations of green/red lights flashed by (D1) and (D2) in all their possible states, is probabilistically defined by:

[1] A particle of type XYZ will cause a detector in state 1 to flash X; a detector in state 2 will flash Y and a detector in state 3 will flash Z.

[2] The emitting source (S) has no knowledge about the states of (D1) and (D2) and there is no communication among the parts of the device.

$$
\begin{aligned}
f(ii) &= XX, \text{ with probability } 1/2, \text{ for } i = 1,2,3, X \in \{G,R\},\\
f(ii) &= XY, \text{ with probability } 0, \text{ for } i = 1,2,3, X,Y \in \{G,R\}, X \neq Y,\\
f(ij) &= XX, \text{ with probability } 1/8, \text{ for } i,j = 1,2,3, i \neq j, X \in \{G,R\},\\
f(ij) &= XY, \text{ with probability } 3/8, \text{ for } i,j = 1,2,3, i \neq j, X,Y \in \{G,R\}, X \neq Y.
\end{aligned}
$$

For example, $f(11) = RR$ with probability $1/2$, $f(11) = GR$ with probability 0, $f(11) = RG$ with probability 0, $f(11) = RR$ with probability $1/2$, $f(12) = GG$ with probability $1/8$, $f(12) = GR$ with probability $3/8$, $f(12) = RG$ with probability $3/8$, $f(12) = RR$ with probability $1/8$, etc.

The automaton transition $\delta : Q \times \Sigma \to Q$ is *not specified.* In fact, varying all transition functions δ we get a class of Mermin automata:

$$\mathcal{M} = (Q, \Sigma, O, \delta, (p_{ij}^{XY}, i,j = 1,2,3, X,Y \in \{G,R\})),$$

where p_{ij}^{XY} is the probability that the automaton on state ij outputs XY: $p_{ii}^{XX} = 1/2$, $p_{ii}^{XY} = 0$, $X \neq Y$, $p_{ij}^{XX} = 1/8$, $p_{ij}^{XY} = 3/8$, $X \neq Y$. This class is fairly large: it contains $9^{18} = 150,094,635,296,999,121$ automata.

4 Computational Complementarity for Probabilistic Automata

In [3] two non-equivalent concepts of computational complementarity were introduced and studied for finite automata. Informally, they can be expressed as follows. Consider the class of all elements of reality (observables) and the following properties:

A Any two distinct elements of reality can be mutually distinguished by a suitably chosen measurement procedure.

B For any element of reality, there exists a measurement which distinguishes between this element and all the others. That is, a distinction between any one of them and all the others is operational.

C There exists a measurement which distinguishes between any two elements of reality. That is, a single pre-defined experiment operationally exists to distinguish between an arbitrary pair of elements of reality.

Clearly, **C** implies **B** and **B** implies **A**, but both converse implications fail to be true; consequently, two *principles of complementarity* emerge:

CI *Property* **A** *but not property* **B** *(and therefore not* **C***):* The elements of reality can be mutually distinguished by experiments, but one of these elements cannot be distinguished from all the other ones by any single experiment.

CII *Property* **B** *but not property* **C***:* Any element of reality can be distinguished from all the other ones by a single experiment, but there does not exist a single experiment which distinguishes between any pair of distinct elements.

We may be regard *CI* as an "uncertainty principle" (cf. [10, p. 21]), later termed "computational complementarity" in [13]. In *CII* each experiment "generates" a pair of distinct states which exercise a mutual influence, namely, they cannot be separated by the experiment; this influence

mimics, in a sense, the state of *quantum entanglement.*[3] We may conceive *CII* as a *toy model* for the *EPR effect* . Under *CII*, for each experiment w we have at least two states q, q' (as distant as we like in terms of the emitting outputs) which interact via the experiment w: any measurement of q is affecting q' and, conversely, any measurement of q' is affecting q.

Motivated by Mermin's automaton probabilistic automaton analysis[4] in [4] computational complementarity has been extended in [5] to a class of probabilistic automata. In opposition with a more popular model, in which the transition is stochastic, but the output is deterministic (see [16]), here we will work with automata having deterministic transitions but stochastic outputs.

Formally, a *finite probabilistic automaton* $\mathcal{A} = (Q, \Sigma, O, \delta, (a_{p,o})_{p\in Q, o\in O})$ consists of an input alphabet Σ, a finite set Q of states, an output finite set O, a transition function $\delta : Q \times \Sigma \to Q$ and an output probabilistic function $f : Q \to O$ given by the matrix $(a_{p,o})_{p\in Q,o\in O}$ satisfying the condition $\sum_{o\in O} a_{p,o} = 1$, for every $p \in Q$. The output emitted by $\mathcal{A}$ on p is $f(p) = o$ with probability $a_{p,o}$. In case of a deterministic finite automaton, for every $p \in Q$, there exists one (unique) output $o \in O$ such that $a_{p,o} = 1$ and all other probabilities are 0; that is, $f(p) = o$. As in case of deterministic automata the transition function extends naturally to $Q \times \Sigma^* \to Q$ satisfying the equation $\delta(p, uv) = \delta(\delta(p, u), v)$, for all $p \in Q, u, v \in \Sigma^*$.[5]

The *response* of the automaton $\mathcal{A}$ on state p to the "experiment" $x = x_1x_2\cdots x_n \in \Sigma^*$ is defined by a concatenation of random variables:

$$Res_{\mathcal{A}}(p, x_1x_2\cdots x_n) = f(p)f(\delta(p, x_1))\cdots f(\delta(p, x_1x_2\cdots x_n)).$$

We say that a state p is *distinguishable* from the state q with *confidence* $\alpha(\alpha \in [1/2, 1])$ if there exists an experiment $x = x_1x_2\cdots x_n \in \Sigma^*$ such that at least one probability that $f(p) \neq f(q)$, $f(\delta(p, x_1x_2\cdots x_i)) \neq f(\delta(q, x_1x_2\cdots x_i)), 1 \leq i \leq n$ is greater or equal to α.

This means that at least on one point, during the "measurement" process of the responses of the automaton to the experiment x, the probability that the response of $\mathcal{A}$ on p and x is different (within the fixed level of confidence) to the response of $\mathcal{A}$ on q and x.[6]

We are now in a position to define properties **A, B, C** for a probabilistic automaton and a level of confidence $\alpha \geq 1/2$.

- A probabilistic automaton has property **A** with level of confidence α if every pair of different states is distinguishable with *confidence* α.
- A probabilistic automaton has **B** with level of confidence α if every state is distinguishable with *confidence* α from any other distinct state.
- A probabilistic automaton has **C** with level of confidence α if there exists an experiment distinguishing with *confidence* α between each pair of different states.

[3] In particular, this influence cannot be used to send an actual message from a state to the other.

[4] Are there two identical, spatially separated, probabilistic automata with identical initial states, whose direct product "simulates" a Mermin's automaton $\mathcal{M}$? The answer is *negative*. In fact, a stronger result is true: *no single state of any Mermin's probabilistic automaton $\mathcal{M}$ can be simulated by the product of the corresponding states of any probabilistic automata $\mathcal{M}_i$.*

[5] In what follows the extension will also be denoted by δ.

[6] The motivation is the following. For a deterministic automaton $\mathcal{B}$, two states p, q are distinguishable if $Res_{\mathcal{B}}(p, x_1 \ldots x_n) \neq Res_{\mathcal{B}}(q, x_1 \ldots x_n)$, for some experiment $x_1 \ldots x_n$ (see [3]), that is, for the corresponding transition and output functions, $f(p) \neq f(q)$ or $f(\delta(p, x_1x_2\cdots x_i)) \neq f(\delta(q, x_1x_2\cdots x_i)), 1 \leq i \leq n$. In the probabilistic case we just replace the above conditions with the corresponding probabilistic ones.

In [5] one shows the existence of probabilistic automata satisfying **A** but not **B** (i.e., *CI*) and **B** but not **C** (i.e., *CII*), respectively.

Properties **A**, **B** and **C** are decidable (cf. [5]). One way to check these properties, with some approximation, is by "simulating" a probabilistic automaton with a deterministic automaton (with some level of confidence). Let $\mathcal{A} = (Q, \Sigma, O, \delta, (a_{p,o})_{p \in Q, o \in O})$ be a probabilistic automaton, and $\alpha \in [1/2, 1]$ a confidence level. We construct a deterministic automaton $\mathcal{A}' = (Q, \Sigma, O', \delta, f')$, where $f : Q \to O'$ is the output function satisfying the following constraints: for every pair of distinct states p, q, if the probability that $f(p) \neq f(q)$ is greater or equal to α, then $f'(p) \neq f'(q)$; otherwise, $f'(p) = f'(q)$.

Note that the above construction cannot be carried on in all cases. For example, consider the probabilistic automaton $\mathcal{A} = (Q, \Sigma, O, \delta, (a_{p,o})_{p \in Q, o \in O})$ where $Q = \{p, q, r\}$, $O = \{G, R\}$, and the output probabilities are $a_{p,G} = 1, a_{p,R} = 0, a_{q,G} = 0, a_{q,R} = 1, a_{r,G} = a_{r,R} = 1/2$ (Σ, δ are arbitrary). It is easy to see that no deterministic automaton $\mathcal{A}'$ simulates $\mathcal{A}$, for every $\alpha > 1/2$. Reason: there is no function f' such that $f'(p) \neq f'(q)$ and $f'(p) = f'(q) = f'(r)$.

If the simulation is possible at the level of confidence α, then *every pair of distinct states $p, q \in Q$ are distinguishable by an experiment applied to $\mathcal{A}$ if and only if they are distinguishable by the same experiment applied to $\mathcal{A}'$*. Consequently, the probabilistic automaton $\mathcal{A}$ has **A** (**B**,**C**) if and only if $\mathcal{A}'$ has **A** (**B**, **C**, respectively).

For example, every Mermin automaton is simulated with confidence $\alpha \in (1/2, 11/16]$ by the deterministic automaton having all components of the Mermin automaton $\mathcal{M}$ and the output function $f'(11) = f'(22) = f'(33) = 0, f'(12) = 1, f'(13) = 2, f'(21) = 3, f'(23) = 4, f'(31) = 5, f'(32) = 6$; if the level of confidence is $\alpha \in (11/16, 7/8]$ we have to change only the output function to $f''(11) = f''(22) = f''(33) = 0, f''(12) = f''(13) = f''(21) = f''(23) = f''(31) = f''(32) = 1$.

5 Mermin's Automata

With confidence 1/2, for every transition function δ, the corresponding Mermin probabilistic automaton has **C**, i.e., it features a classical behaviour. Indeed,

- for every $i \neq j$, the probability that $f(ii) \neq f(jj) = 1/2$,
- for every $j \neq k$, the probability that $f(ii) \neq f(jk) = 7/8$,
- for every $i \neq j, k \neq l$, $ij \neq kl$, the probability that $f(ij) \neq f(kl) = 11/16$.

Is the above result true when the simulation is done at higher levels of confidence? Two such intervals recommend themselves from the construction of Mermin deterministic automata: $(1/2, 11/16]$ and $(11/16, 7/8]$. In what follows we will be interested in checking the distribution of properties **C**, *CI*, and *CII* for two classes of deterministic automata simulating Mermin deterministic automata with confidence levels in the above intervals. For the first interval of confidence we consider all automata keeping all components of a Mermin probabilistic automaton and the output function f'; in the second case we use the output function f'' (see Section 4). More precisely, our automata will be characterized by $Q = \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$, $\Sigma = \{G, R\}$, and the output functions $f' : Q \to O'$, respectively, $f'' : Q \to O'$, where $O' = \{0, 1, 2, 3, 4, 5, 6\}$, and $O'' = \{0, 1\}$.

6 Sample Space, Confidence and Errors

As it is easy to see, varying all transition functions we get a population consisting of 9^{18} automata. With such a large a population it is difficult to test properties **C**, *CI*, and *CII* for *all* automata. Consequently, we will be looking at simulations. We will briefly present the method (see, for instance, [8], Chapter 4).

Given a finite population of size N, we will estimate 3 proportions associated with 3 binary random variables,

$$P_i = P\left(X_i = 1\right), i = 1, 2, 3,$$

using a pseudo-random sample of size n. Each estimate should be correct within $\pm c\%$ in the sense that, if the sample shows p_i to be P_i, the percentage for the whole population is "sure" to lie between $p_i - c\%$ and $p_i + c\%$ (with "accuracy within $c\%$"). As we cannot absolutely guarantee an accuracy within $c\%$, we accept a probability α (0.0027 in our case) of getting "an unlucky sample" (which is in error by more than the desired $c\%$).

Random generation of the sample is equivalent with sampling with replacement (generated units are independent, repetitions are possible), according to a Binomial scheme. If $(u_1, ..., u_n)$ are generated items, we denote by m_i the number of items for which $X_i(u) = 1$, $i = 1, 2, 3$. Then, the estimates of P_i are

$$p_i = \frac{m_i}{n}, i = 1, 2, 3.$$

For a large value of n ($n > 100$), one can use the normal approximation for p_i, that is, p_i is approximately normal distributed $N\left(P_i, \frac{P_i(1-P_i)}{n}\right)$.

In order to estimate the value of n, we start from the simultaneous conditions

$$\Pr\left(\mid p_i - P_i \mid \geq z_{1-\frac{\alpha}{2}} \sqrt{Var\left(p_i\right)}\right) = \alpha, i = 1, 2, 3,$$

where $z_{1-\frac{\alpha}{2}}$ is the $(1 - \frac{\alpha}{2})$–quantile of the $N(0, 1)$ distribution, and $z_{0.99865} \approx 3$ (see, for instance, [14], Table II).

Accordingly, we will have

$$z_{1-\frac{\alpha}{2}} \sqrt{\frac{P_i\left(1 - P_i\right)}{n}} = c, i = 1, 2, 3,$$

hence, the sample size should be equal to

$$n = \max\left\{\frac{z^2_{1-\frac{\alpha}{2}} P_i\left(1 - P_i\right)}{c^2}, i = 1, 2, 3\right\}.$$

Notice that this value of n depends on the unknown proportions P_i. Therefore, we need a prior statement on the range of each P_i and, to be on the safe size, we will choose P_i equal to the value that maximizes the product $P_i(1 - P_i)$. In the absence of such prior knowledge regarding P_i, we will choose the "critical" value $P = 50\%$ (which maximizes the product $P(1 - P)$). Hence, the "safest" estimation of the sample size n is

$$\hat{n} = \frac{z^2_{1-\frac{\alpha}{2}} \cdot 2,500}{c^2}.$$

For our level of significance $\alpha = 0.0027$, one gets $z_{1-\frac{\alpha}{2}} = 3$. For an accuracy within $c = 2\%$, $n = 3,725$, which is the size of the sample investigated by our programs.

Because of the large population of automata, the same value $\hat{n}$ can be used even in case of dependent generated units. Indeed, a pseudo-random generation process with dependent units is

equivalent with sampling without replacement (the hypergeometric scheme). In this case, by a similar computation we get

$$n = \max\left\{\left(\frac{z_{1-\frac{\alpha}{2}}2P_i\left(1-P_i\right)}{c^2}\right) / \left(1+\frac{1}{N}\left(\frac{z_{1-\frac{\alpha}{2}}2P_i\left(1-P_i\right)}{c^2}-1\right)\right), i=1,2,3\right\}.$$

For a very large population (as in our case, $N = 9^{18}$), the denominator is approximately equal to 1, as

$$\frac{1}{N}\left(\frac{z^2_{1-\frac{\alpha}{2}}P_i\left(1-P_i\right)}{c^2}-1\right) \approx 0,$$

and, again, we find the same value for $\hat{n}$.

An alternative approach based on *confidence intervals* [9] was considered and actually used for cross-checking the estimated probabilities. Given a *confidence level* α, the bounds L_i, U_i of one of our confidence intervals for P_i are given by the roots of the following two equations (where Φ is the normal repartition function $N(0,1)$):

$$1-\Phi\left(\frac{m_i - n_i\cdot L_i}{\sqrt{n_i\cdot L_i\cdot(1-L_i)}}\right) = \alpha/2,\ \Phi\left(\frac{m_i - n_i\cdot U_i}{\sqrt{n_i\cdot U_i\cdot(1-U_i)}}\right) = \alpha/2.$$

The results are statistically sound and further described in [7].

7 Codification and Programs

This section briefly describes our programs.[7] At the core we have a data structure to represent an automaton, and three routines to test properties **A**, **B**, **C**. We chose a simple array based structure which enables both fast processing in different languages and is readable by humans. Specifically, the string "122210" represents an automaton $\mathcal{A}$ with 2 states $Q = \{1,2\}$, 2 input symbols $\Sigma = \{0,1\}$, 2 output symbols $O = \{0,1\}$, a transition function $\delta : Q \times \Sigma \to Q$, defined by $\delta(1,0) = 1$, $\delta(1,1) = 2$, $\delta(2,0) = 2$, $\delta(2,1) = 2$, and an output function $f : \Sigma \to O$, defined by $f(1) = 1$, $f(2) = 0$.

More generally, consider an automaton $\mathcal{A}$ with n states $Q = \{1,2,\ldots,n\}$, i input symbols $\Sigma = \{0,1,\ldots,i-1\}$, and j output symbols $O = \{0,1,\ldots,j-1\}$. This automaton $\mathcal{A}$ can be represented as an array (string) of $n\cdot i + n$ symbols, of which the first $n\cdot i$ are state numbers (corresponding to the $n\cdot i$ values of the transition function δ) and the last n are output symbols (corresponding to the n values of the output function f). This representation is unique with respect to the three essential automaton parameters (n,i,j). These automata representations can be sorted lexicographically, and then assigned unique sequence indices. Thus we have a bijective mapping between this set of automata and non-negative integers in the interval $[0, n^{n\cdot i}\cdot j^n - 1]$. For example, all automata with 2 states, 2 input symbols, and 2 output symbols can be ordered in the following sequence:

$$0 \leftrightarrow 111100, 1 \leftrightarrow 111101, 2 \leftrightarrow 111110, 3 \leftrightarrow 111111, 4 \leftrightarrow 111200, 5 \leftrightarrow 111201, \ldots, 63 \leftrightarrow 222211,$$

and therefore each such automaton uniquely corresponds to a non-negative integer in the interval $[0, 2^4\cdot 2^2 - 1] = [0,63]$.

Ignoring or fixing the output function f, we get a more restricted range, i.e., we have a bijective mapping between automata with n states and i input symbols and non-negative integers in $[0, n^{n\cdot i}-$

[7] More details about our data structures and algorithms can be found in [6, 7].

1]. For example, the automata with 2 states and 2 input symbols correspond to the first 16 non-negative integers, and the automata with 9 states and 2 input symbols correspond to the first 9^{18} non-negative integers.

We use this second mapping to generate pseudo-random automata with a given fixed output. For example, to generate a pseudo-random deterministic Mermin automaton we compute a pseudo-random `long` non-negative integer in $[0, 9^{18}-1]$, convert this number into its corresponding automaton representation, and finally append the required output function, i.e., we append the string "000123456" for f' and "000111111" for f''. However, appended strings need not match one of these two forms exactly as the order in which our states or output symbols are numbered is irrelevant here. Thus, the above string "000111111" for f'' could well be replaced by any combination of 9 occurrences of 0 and 1, as long as we have 3 occurrences of one digit and 6 of the other. For example, instead of "000111111" we can use either of "111111000", "110110110", "000111000", or "000000111", or any of the $X = 168$ distinct $0-1$ patterns which can be "converted" to "000111111" by re-numbering the states and/or the output symbols.[8] In fact, for internal reasons (of combinatorial nature) we prefer to use "000000111" for representing the output function f''.

Our arguments can be extended to show that we can restrict our study of 9 state automata to 30 "essential" output strings, and all other cases can be converted to one of these by means of permutations (of states) and re-labeling (of output symbols). Here are our 30 "essential" output strings for 9 state automata: "000000000", "000000001", "000000011", "000000111", "000001111", "000000012", "000000112", "000001112", "000011112", "000001122", "000011122", "000111222", "000000123", "000001123", "000011123", "000011223", "000111223", "000112233", "000001234", "000011234", "000111234", "000112234", "001122334", "000012345", "000112345", "001122345", "000123456", "001123456", "001234567", "012345678". These figures may help understanding the context in which Mermin automata appear: they correspond to "000000111" and "000123456".

Our `PropertyA` routine takes an automaton $\mathcal{A}$ as a parameter and returns the Boolean value `true` if $\mathcal{A}$ has the property **A**, and the value `false` otherwise. Essentially, this routine consists of three embedded loops: an outer loop iterating over the states, a middle loop iterating again over the states, and an inner loop iterating over a subset of words $W \subset \Sigma^*$. Its purpose is to determine the value of the predicate $\forall p \in Q, \forall q \in Q \setminus \{p\}, \exists w \in W, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)$. Here follows a high-level pseudo-code, without optimizations.

```
boolean PropertyA(Automaton A) begin
    for each p ∈ Q do
        for each q ∈ Q \ {p} do
            for each w ∈ W do
                if Res_A(p,w) ≠ Res_A(q,w) then next q
            end for w
            return false
        end for q
    end for p
    return true
end PropertyA
```

The codes for `PropertyB` and `PropertyC` routines are similar, using the same three loops but in different order. The `PropertyB` routine contains an outer loop on the states, a middle loop on the words, and an inner loop on the states again. Its purpose is to determine the value of the predicate $\forall p \in Q, \exists w \in W, \forall q \in Q \setminus \{p\}, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)$. The `PropertyC` routine contains an outer

[8] In fact, these "conversions" are part of a more general concept of *isomorphism* between Moore automata, but we don't follow this path in this paper; here we just count automata *definitions*, not *equivalence classes*.

loop on the words, a middle loop on the states, and an inner loop on the states again. Its purpose is to determine the value of the predicate $\exists w \in W, \forall p \in Q, \forall q \in Q \setminus \{p\}, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)$.

Several important details must be worked out before actually running this code:

- How many strings should we have in the set W? The theory [3, 6] doesn't give complete answers for the upper bounds on the lengths of the strings which must be tested: we know that n symbols are enough to test the property **A**, but there is no tight bound on the number of symbols which must be tested for properties **B** and **C**. It has been estimated that 2^n are enough, but no example is known that requires more then $2n-5$ symbols for these properties. Moreover, extensive exploration showed us that such "pathological" cases which would require more than $2n-5$ symbols are extremely rare, if any. Indeed, even some 7 letters seem enough to differentiate between the properties **A**, **B**, and **C** for most of the 9 state automata. Therefore, for all practical purposes we decided to use the empirical limits $n+3$ for the property **B** and $2n-5$ for the property **C**. Further work should (dis)prove that this assumption is right for estimating statistical data.

- In which order should we process the words of W? A random order seems to give faster results. However, we proceed in a systematic way, exhausting in order the words of length $0, 1, 2, \ldots, n$. This process is a bit slower but it enables us to collect histograms on the minimal lengths of words needed to differentiate between our properties, thus validating our assumptions on the practical upper bounds. We maintain a separate histogram for each of the properties **A**, **B**, **C**, where each automaton occurs exactly once in each histogram. Specifically, for each automaton $\mathcal{A}$, the histogram for property **A** has one entry for $\min\{l \in \mathbf{N} \mid \forall p \in Q, \forall q \in Q \setminus \{p\}, \exists w \in W, |w| \leq l, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)\}$, the histogram for property **B** has one entry for $\min\{l \in \mathbf{N} \mid \forall p \in Q, \exists w \in W, \forall q \in Q \setminus \{p\}, |w| \leq l, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)\}$, and the histogram for property **C** has one entry for $\min\{l \in \mathbf{N} \mid \exists w \in W, \forall p \in Q, \forall q \in Q \setminus \{p\}, |w| \leq l, Res_{\mathcal{A}}(p,w) \neq Res_{\mathcal{A}}(q,w)\}$.

- How can we further optimize this code? The above sketched implementation for `PropertyA` has a complexity of $O(n^2 \cdot 2^{n+1} \cdot n)$, because in the innermost line we have to consider up to n^2 pairs of states and 2^{n+1} words of length of up to n symbols each. There are some obvious optimizations, such as avoiding to test the pair (q,p) if the pair (p,q) was already tested. Theoretically, the algorithms can be substantially improved by using *tree-like* representations of the words in W, or *dynamic programming* techniques. However, such more "sophisticated" algorithms have an inherent overhead which pays off asymptotically for large inputs, but which doesn't pay off well when most of the differentiating words have short lengths (of up to around 7 symbols). Further work is expected in this direction.

8 Results

We have collected statistics for samples of size 3725, using both sampling methods discussed in Section 6, i.e., with and without replacement. As expected, both methods gave the same results and indeed, the chance of generating two identical automata from a large population of some $150 \cdot 10^{15}$ automata is practically zero.

Here follow the main results of a typical run, a table giving the distribution of Mermin automata in classes **C**, *CI*, *CII*, and the rest (here called *None*). The first row corresponds to deterministic Moore automata with 9 states and output pattern "000123456" which simulate Mermin probabilistic automata with confidence $\alpha \in (1/2, 11/16]$. The second row corresponds to deterministic Moore automata with 9 states and the output pattern "000000111" which simulate Mermin probabilistic automata with confidence $\alpha \in (11/16, 7/8]$. The observed frequencies differ slightly between different runs, but probabilities are determined with an accuracy of $c = 2\%$.

Output pattern	**C**	*CI*	*CII*	*None*	**C%**	*CI%*	*CII%*	*None%*
000123456	3280	211	6	228	88.05%	5.66%	0.16%	6.12%
000000111	436	1601	504	1184	11.70%	42.98%	13.53%	31.79%

For comparison we present a table giving the distribution of all Moore automata with 9 states and 2, 7 and 9 output symbols in the classes **C**, *CI*, *CII*, and *None*.[9]

Output size	**C**	*CI*	*CII*	*None*	**C%**	*CI%*	*CII%*	*None %*
2	526	1583	484	1132	14.12%	42.50%	12.99%	30.39%
7	2839	316	260	310	76.21%	8.48%	6.98%	8.32%
9	3028	185	206	306	81.29%	4.97%	5.53%	8.21%

The above results show that at any level of confidence $\alpha \in (1/2, 11/16]$, the class of deterministic automata simulating Mermin probabilistic automata display less correlations than typical deterministic automata with 9 states and 7 outputs, but at higher levels of confidence $\alpha \in (11/16, 7/8]$, when the simulation is more accurate, *deterministic automata simulating Mermin probabilistic automata display more correlations than typical deterministic automata with 9 states and 2 outputs.* In the last case, *EPR correlations established in* [4] *for Mermin probabilistic automata correspond to computational complementarity of the deterministic automata simulating Mermin probabilistic automata,* [10, 13, 18, 3, 6].

References

[1] Aspect, A. Bell's inequality test: more ideal than ever, *Nature* 398 (1999), 189–190.

[2] Bell, J. S. *Speakable and Unspeakable in Quantum Mechanics*, Cambridge University Press, Cambridge, 1987.

[3] Calude, C., Calude, E., Svozil, K. and Yu, S. Physical versus computational complementarity I, *International Journal of Theoretical Physics, 36* (1997), 1495–1523.

[4] Calude, C. S., Calude, E. and Svozil, K. Quantum correlations conundrum: An automaton-theoretic approach, *Proceedings of WIA'99,* Potsdam, Germany, 1999, in press.

[5] Calude, C. S., Calude, E. and Svozil, K. Computational complementarity for probabilistic automata, in C. Vide, V. Mitrana (eds.). *Words, Sequences, Languages: Where Computer Science, Biology and Linguistics Meet*, Kluwer, Dordrecht, 2000, in press.

[6] Calude, E. *Automata-Theoretic Models for Computational Complementarity,* Ph. D. Thesis, Auckland University, New Zealand, 1998.

[7] Chiou, T. *Testing Computational Complementarity For Finite Automata Using Distributed Object Technology,* MSc Thesis, Auckland University, New Zealand, 1999, in progress.

[8] Cochran, W. G. *Sampling Techniques,* Third edition, John Wiley, New York, 1977.

[9] Collani, E. von, Dräger, K., Hottendorf, J. *Tables for Optimal Two-Sided Confidence Intervals and Tests for un Unknown Probability.* Monograph Series in Stochastics, 1, Institut für Angewandte Mathematik und Statistik der Universität Würzburg, Würzburg, 1996.

[9] We made this choice because our deterministic simulations of Mermin automata use either 2 or 7 output symbols, and 9 is the maximum number of output symbols of a Moore automaton with 9 states. In [7] one presents an extensive statistics of all Moore automata with 9 states and 9 output symbols, corresponding to each of the 30 essential output patterns introduced in Section 7.

[10] Conway, J. H. *Regular Algebra and Finite Machines.* Chapman and Hall Ltd., London, 1971.

[11] Dürr, S., Nonn, T., and Rempe, G. Origin of quantum mechanical complementarity probed by a 'which-way' experiment in an atom interferometer, *Nature* 395 (1998), 33.

[12] Einstein, A., Podolsky, B., and Rosen, N. Can quantum-mechanical description of physical reality be considered complete? *Physical Review 47* (1935), 777–780. Reprinted in [19, pp. 138–141].

[13] Finkelstein, D., and Finkelstein, S. R. Computational complementarity. *International Journal of Theoretical Physics 22*, 8 (1983), 753–779.

[14] Hald, A. Statistical Tables and Formulas, John Wiley, New York, 1965.

[15] Mermin, N. D. Bringing home the atomic world: Quantum mysteries for anybody. *American Journal of Physics* 49 (1981), 940–943.

[16] Paz, A. *Introduction to Probabilistic Automata*, Academic Press, New York, 1971.

[17] Peres, A. *Quantum Theory: Concepts and Methods*, Kluwer Academic Publishers, Dordrecht, 1993.

[18] Svozil, K. *Randomness & Undecidability in Physics.* World Scientific, Singapore, 1993.

[19] Wheeler, J. A., and Zurek, W. H. *Quantum Theory and Measurement.* Princeton University Press, Princeton, 1983.