Hidden variables simulating quantum contextuality increasingly violate the Holevo bound

Adán Cabello[1] and Joost J. Joosten[2]

[1] Departamento de Física Aplicada II
Universidad de Sevilla, 41012 Sevilla, Spain
adan@us.es
www.adancabello.com

[2] Dept. Lògica, Història i Filosofia de la Ciència
Universitat de Barcelona, Montalegre 6, 08001 Barcelona, Spain
jjoosten@ub.edu
www.phil.uu.nl/∼jjoosten/

Abstract. In this paper we approach some questions about quantum contextuality with tools from formal logic. In particular, we consider an experiment associated with the Peres-Mermin square. The language of all possible sequences of outcomes of the experiment is classified in the Chomsky hierarchy and seen to be a regular language.

Next, we make the rather evident observation that a finite set of hidden finite valued variables can never account for indeterminism in an ideally isolated repeatable experiment. We see that, when the language of possible outcomes of the experiment is regular, as is the case with the Peres-Mermin square, the amount of binary-valued hidden variables needed to de-randomize the model for all sequences of experiments up to length n grows as bad as it could be: linearly in n.

We introduce a very abstract model of machine that simulates nature in a particular sense. A lower-bound on the number of memory states of such machines is proved if they were to simulate the experiment that corresponds to the Peres-Mermin square. Moreover, the proof of this lower bound is seen to scale to a certain generalization of the Peres-Mermin square. For this scaled experiment it is seen that the Holevo bound is violated and that the degree of violation increases uniformly.

1 Introduction

In this paper we will focus on an experiment that is associated to the famous Peres-Mermin square [8, 9]. The experiment consists of a sequence of measurements performed consecutively on a two-qubit system. All the measurements are randomly chosen from a subset of those represented by two-fold tensor products of the Pauli matrices X, Y and Z, and the identity $\mathbb{I}$. The set $\mathcal{L}$ of all sequences of outcomes consistent with Quantum Mechanics is studied as a formal language.

In the theory of formal languages, the Chomsky hierarchy [4, 10] defines a classification of languages according to their level of complexity. In Section 2, this language $\mathcal{L}$ will be classified in the Chomsky hierarchy. It will be seen to live in the lower regions of the hierarchy. More concrete, it will be seen that the language is of Type 3, also called *regular*.

In Subsection 3.1, the rather evident observation is made that a finite set of hidden finite valued variables can never account for indeterminism in an ideally isolated repeatable experiment.

Finally, in Subsection 3.2, the question is addressed how much memory is needed to simulate Quantum Mechanics in experiments with sequential measurements. The question naturally arises: *how* are we allowed to simulate nature. We wish to refrain from technical implementation details

of these simulations as much as possible. To this extent, we invoke the Church-Turing thesis that captures and mathematically defines the intuitive notion of what is computable at all by what mechanized and controlled means so-ever. This gives rise to our notion of MAGAs: Memory-factored Abstract Generating Automata. We prove a lower bound for the amount of memory needed for MAGAs that simulate extensions of the experiment associated to the Peres-Mermin square and shall see that this bound directly and increasingly so violates the Holevo bound.

2 Language defined by the experiment

In this paper we will denote the Peres-Mermin square by the following matrix

$$\begin{pmatrix} A & B & C \\ a & b & c \\ \alpha & \beta & \gamma \end{pmatrix} \qquad (\dagger),$$

where these variables can get assigned values in $\{1, -1\}$. The corresponding –classically impossible to satisfy– restriction is that the product of any row or column should be 1 except for C, c, γ which should multiply to -1.

Basically, contextuality amounts to the phenomenon that the outcome of a measurement on a system relates to and depends on other (compatible) measurements performed on that system.

In the next Subsection 2.1, we will first formally describe the possible outcomes of the experiment that corresponds to the Peres-Mermin square. We will describe this in almost tedious detail as we later need to formalize the corresponding language.

2.1 The experiment

We will collect the nine observables of our experiment into an alphabet Σ which we denote by

$$\Sigma \quad := \quad \{A, B, C, a, b, c, \alpha, \beta, \gamma\}.$$

The experiment consists of arbitrarily many discrete consecutive measurements of these nine observables which can take values in the two-element set $\{-1, 1\}$. For reference we reiterate that (†) in this paper coincides with the well-studied Peres-Mermin square [8, 9].

Definition 1 (Context; Compatible observables). *The rows and columns of matrix* (†) *are called* contexts. *Two observables within the same context are called* compatible *and two observables that do not share a common context are called* incompatible.

It is clear that each observable belongs to exactly two contexts. Likewise, each observable is compatible with 4 other observables and incompatible with 4 yet other observables. Now, let us define what it means for an observable to be *determined*.

Definition 2 (Determined observables; Value of a determined observable). *An observable becomes (or stays)* **determined** *if:*

(E1) Once we measure an observable, it becomes (or stays) determined and its value is the value that is measured, either 1 or -1. If the observable X that was measured was already determined, then the value of X that is measured anew must be the same as in the most recent measurement.

(E2) If two observables within one context are determined, and if the third observable in this context was not yet determined, this third value becomes determined too. Its corresponding value is such that the product of the three determined values in this context equals 1. The sole exception to this value assignment is the context $\{C, c, \gamma\}$ that should multiply to -1.

The notion of an observable being undetermined *is defined by the following clause:*

(D1) By default, an observable is **undetermined** *and only becomes determined in virtue of (E1) or (E2). An observable X that is determined remains determined* **if and only if** *all successive measurements are in one of the two contexts of X that is, all successive measurements are compatible with X. As soon as, according to this criterion, an observable is no longer determined we say that it has become* **undetermined**. *Undetermined observables stay undetermined until they become determined. Undetermined observables have no value assigned. Sometimes we will say that the value of an undetermined observable is* undefined.

A sequence of measurements is *consistent* with our experiment if its determined observables meet the restrictions above. In essence, part of our definition is of inductive nature. To see how this works, let us see this, by way of example, the sequence of measurements $[A = 1; B = 1; c = 1; \gamma = 1]$ is inconsistent with our experiment:

Measured observable	Measured value	Comments
A	1	The experiments starts so by default, all observables were undetermined (D1). After the measurement, by (E1) the observable A is determined and assigned the value 1.
B	1	As B is a new measurement, the observable becomes determined (E1) with value 1. The observable A remains determined by ($D1$). Moreover, the observable C which is in the context $\{A, B, C\}$, now becomes determined in virtue of (E2) with value 1 too as the product $A \cdot B \cdot C$ should multiply to 1.
c	1	By (E1), c becomes determined with value 1 and the observables A and B become undetermined in virtue of ($D1$). The observable C remains determined by (D1) with value 1 as the new measurement of c is in the context $\{C, c, \gamma\}$. Thus, in virtue of (E2), the observable γ becomes determined too. Its value must be -1 as $C \cdot c \cdot \gamma$ should multiply to -1.
γ	1	As we said before, in virtue of ($E2$) the value of γ should be -1. Thus this is inconsistent with the measurement of 1 for γ.

Note that only the last measurement was inconsistent with our experiment.

Remark 1. If we fill the square (†) with any assignment of 1s and -1s, then the number of products of contexts that equal -1 will always be even.

Proof. By induction on the number of -1s. If all observables in (†) are set to one, then all contexts multiply to one thus yielding zero –an even number– of negative products. If we add

one more -1 to (†), this -1 will occur in exactly two contexts thereby flipping the sign of the respective products of these two contexts.

2.2 The formal language

In this subsection we shall specify a formal language comprising exactly the possible strings of measurements of our above specified experiment. We mention where the complexity of this language resides in the Chomsky hierarchy: the regular languages. Let us first briefly introduce some terms and definitions from the theory of formal languages.

We shall call a collection of symbols an *alphabet* and commonly denote this by Σ. A *string* or *word* over an alphabet is any finite sequence of elements of Σ in whatever order. We call the sequence of length zero the *empty string*, and will denote the empty string/word by λ. We will denote the set of all strings over Σ by Σ^* using the so-called Kleene-star. Thus, formally and without recurring to the notion of sequence, we can define Σ^*, the set of all finite strings over the alphabet Σ as follows.

$$\lambda \in \Sigma^*;$$
$$\sigma \in \Sigma^* \ \& \ s \in \Sigma \ \Rightarrow \ \sigma s \in \Sigma^*.$$

Instead of writing $\lambda\sigma$, we shall just write σ. It is clear that Σ^* is an inductive definition so that we also have an induction principle to prove or define properties over Σ^*. For example, we can now formally define what it means to *concatenate* – stick the one after the other– two strings: We define $\star$ to be the binary operation on Σ^* by $\sigma \star \lambda = \sigma$ and $\sigma \star (\tau s) = (\sigma \star \tau)s$. Any subset of Σ^* is called a *language* over Σ.

The study of formal languages concerns, among others, which kind of grammars define which kind of languages, and by what kind of machines these languages are recognized. In the current paper we only need to provide a formal definition of so-called *regular* languages. We do this by employing regular grammars. Basically such a grammar is a set of rules that tells you how strings in the language can be generated.

Definition 3 (Regular Grammar). *A regular grammar over an alphabet Σ consist of a set $\mathcal{G}$ of* generating symbols *together with a set of* rules*. In this paper we shall refer to generating symbols by using a line over the symbols. The generating symbols always contain the special start-symbol $\overline{S}$. Rules are of the form*

$$\overline{X} \to \lambda \quad \textit{or}$$
$$\overline{X} \to s\overline{Y},$$

where $\overline{X}, \overline{Y} \in \mathcal{G}$ and $s \in \Sigma$. The only restriction on the rules is, that there must be at least one rule where the left-hand side is $\overline{S}$.

Informally, we state that a *derivation* in a grammar is given by repeatedly applying possible rules starting with $\overline{S}$, where the rules can be applied within a context. Thus, for example, when we apply the rule $\overline{A} \to a\overline{B}$ in the context σA, we obtain $\sigma a\overline{B}$. A more detailed example of a derivation is given immediately after Definition 4. We say that a string σ over Σ is derivable within a certain grammar if there is a derivation resulting in σ. The *language defined by a grammar* is the set of derivable strings over Σ^*. A language is called *regular* if it is definable by a regular grammar.

We are now ready to give a definition of a regular language that, as we shall see, exactly captures the outcomes of our experiment. To this end, we must resort to a richer language than just Σ as Σ only comprises the observables and says nothing over the outcomes. So, we shall consider a language where, for example, $\tilde{A}$ will stand for, "A was measured with value -1", and

A will stand for, "A was measured with value 1". We will denote this alphabet by $\tilde{\Sigma}$. We will use the words *compatible* and *incompatible* in a similar fashion for $\tilde{\Sigma}$ as we did for our observables in Σ. Thus, for example, we say that both B and $\tilde{B}$ are compatible with C. The only difference will be that $\tilde{A}$ is not compatible with A whereas A is.

Definition 4 (A grammar for $\mathcal{L}$). *The language $\mathcal{L}$ will be a language over the alphabet $\tilde{\Sigma} := \{A, \tilde{A}, B, \tilde{B}, \ldots, \beta, \tilde{\beta}, \gamma, \tilde{\gamma}\}$, where the intended reading of A will be that the observable A was measured to be 1, and $\tilde{A}$ will stand for measuring -1, etc.*

Before we specify the grammar that will generate $\mathcal{L}$, we first need some notational conventions. In the sequel, U, V, X, Y and Z will stand for possible elements of our alphabet. If X and Y are compatible symbols, we will denote by $Z(XY)$ the unique symbol that is determined in (E2) by X and Y. Thus, for example, $Z(A\tilde{B}) = \tilde{C}$ and $Z(C\gamma) = \tilde{c}$.

The generating symbols of the grammar will be denoted by a string with a line over it. Note that, for example, $\overline{XY}$ is regarded as one single generating symbol. The intended reading of such a string is that the two symbols are different and compatible, the last symbol is the one that can be generated next, and the remainder of the string codifies the relevant history. As usual we will denote the initial generating symbol by $\overline{S}$. Let $\mathcal{L}$ be the formal language generated by the following grammar.

$\overline{S}$	$\longrightarrow$	λ	
$\overline{S}$	$\longrightarrow$	$\overline{X}$	*for any symbol $X \in \Sigma$*
$\overline{X}$	$\longrightarrow$	X	
$\overline{X}$	$\longrightarrow$	$X\overline{X}$	
$\overline{X}$	$\longrightarrow$	$X\overline{Z}$	*for Z incompatible with X*
$\overline{X}$	$\longrightarrow$	$X\overline{XY}$	*for Y compatible with X (but not equal)*
$\overline{XY}$	$\longrightarrow$	Y	
$\overline{XY}$	$\longrightarrow$	$Y\overline{XY}$	
$\overline{XY}$	$\longrightarrow$	$Y\overline{YX}$	
$\overline{XY}$	$\longrightarrow$	$Y\overline{YZ}$	*for Z compatible with Y (not equal), but not with X*
$\overline{XY}$	$\longrightarrow$	$Y\overline{YZ(XY)}$	
$\overline{XY}$	$\longrightarrow$	$Y\overline{Z(XY)U}$	*for U compatible with $Z(XY)$ (but not equal) but not compatible with X or Y*

We emphasize that the conditions on the right are not part of the rules. Rather, they indicate how many rules of this type are included in the grammar. For example, $\overline{S} \longrightarrow \overline{X}$ for any symbol $X \in \Sigma$ is our short-hand notation for nine rules of this kind.

Let us give an example of how this grammar works to the effect that $ABc\tilde{\gamma}$ is in our language. Recall our reading convention that says that A stands for measuring $A = 1$, B for measuring

$B = 1$, c for $c = 1$, and $\tilde{\gamma}$ for measuring $\gamma = -1$. Here goes a derivation of the string $ABc\tilde{\gamma}$:

String derived	**Instantiation of rule**	**General rule applied**
$\overline{A}$	$\overline{S} \longrightarrow \overline{A}$	By the rule $\overline{S} \longrightarrow \overline{X}$ with $X = A$
$A\overline{AB}$	$\overline{A} \longrightarrow A\overline{AB}$	By the rule $\overline{X} \longrightarrow X\overline{XY}$ (X and Y compatible) with $X = A$, $Y = B$. Note that A and B are indeed compatible.
$AB\overline{Cc}$	$\overline{AB} \longrightarrow B\overline{Cc}$	By the rule $\overline{XY} \longrightarrow Y\overline{Z(XY)U}$ for U compatible with $Z(XY)$ (but not equal), but not compatible with X or Y, where $X = A$, $Y = B$, $Z(XY) = C$ and $U = c$.
$ABcc\overline{\tilde{\gamma}}$	$\overline{Cc} \longrightarrow c\overline{c\tilde{\gamma}}$	By the rule $\overline{XY} \longrightarrow Y\overline{YZ(XY)}$ with $X = C$, $Y = c$ and $Z = \tilde{\gamma}$.
$ABc\tilde{\gamma}$	$\overline{c\tilde{\gamma}} \longrightarrow \tilde{\gamma}$	By the rule $\overline{XY} \longrightarrow Y$ with $X = c$ and $Y = \tilde{\gamma}$.

In a previous example in Subsection 2.1, we showed that the string $ABC\gamma \in \tilde{\Sigma}$ is not consistent with the experiment. It is not hard to prove that in our grammar there is no derivation of this string either.

We note and observe that the grammar has various desirable properties. As such, the grammar is monotone[3], where each generated string contains at most one generating symbol. Moreover, it is easily seen that once a string generated by the grammar contains a composite generating symbol (like $\overline{XY}$), each subsequently generated string will also contain a composite generating symbol if it contains any generating symbols at all. Indeed, the grammar is very simple.

Theorem 1. *The language $\mathcal{L}$ as defined in Definition 4 coincides with the set of consistent measurements as defined in Definition 2.*

Proof. We must show that, on the one hand any string in $\mathcal{L}$ is consistent with the experiment, and on the other hand, any sequence of measurements consistent with the experiment is derivable in $\mathcal{L}$.

The first implication is proven by an induction on the length of the sequence of measurements. The fact that not measuring at all is consistent is reflected by $\lambda \in \mathcal{L}$. For non-empty sequence we distinguish between a context being determined or not. The first case is covered by rules of the form $\overline{XY} \rightarrow$ righthand side. Note that in the experiment, once there is a context fully determined, in any future measurement there will be a (possibly different) context that is fully determined. This is reflected in the grammar in the sense, that once a composite generating symbol (of the form $\overline{XY}$) enters the derivation, in all subsequent derivations all generating symbols are composite. In this sense, the composite generating symbols correspond exactly to the case where there is a context fully determined. Note that for any consistent new measurement there is a corresponding rule (righthand side). The second case is easier and corresponds to all rules of the form $\overline{X} \rightarrow$ righthand side.

A proof of the second implication proceeds by induction on the length of a derivation. We first note that any non-empty string in $\tilde{\Sigma}^*$ that is derivable in $\mathcal{L}$ is of the form $\sigma \star \Xi$ where $\sigma \in \tilde{\Sigma}^*$ and $\Xi \in \mathcal{G}$. Although each $\Xi \in \mathcal{G}$ is a single separate symbol we already suggestively had composite notations like $\overline{XY}$ for them. We define $l(\Xi)$ to be "the last symbol" of Ξ so that $l(\overline{S}) = \lambda$,

[3] That is, the length of each subsequent string in a derivation is at least as long as the length of the previous one.

$l(\overline{X}) = X$, and $l(\overline{XY}) = Y$. We now can prove by an easy induction on the length of a derivation in $\mathcal{L}$, that for any string $\sigma\Xi$ that is derivable, the corresponding string of measurements $\sigma l(\Xi)$ is consistent with the experiment. Again we use here the distinction between composite generating symbols and non-composite generating symbols and their correspondence to a context being fully determined or not.

Note that by the mere syntactic properties of the definition of $\mathcal{L}$ we see that $\mathcal{L}$ is indeed a regular language.

Corollary 1. *The set of consistent sequences of measurements of the Peres-Mermin experiment is a regular language.*

Regular languages are of Type 3 in the Chomsky hierarchy. As such we have access to a corpus of existing theory. In particular, there exists a method to determine the minimal amount of states in a Deterministic Finite State Automata that will accept $\mathcal{L}$. Moreover, we also have access to the following proposition [5].

Proposition 1. *Let $\mathcal{L}$ be a regular language. Then, there exist polynomials $p_1, \ldots, p_k$ and constants $\lambda_1, \ldots, \lambda_k$ such that the number of strings of length n in $\mathcal{L}$ is given by*

$$p_1(n)\lambda_1^n + \ldots + p_k(n)\lambda_k^n.$$

As we have seen already, now that we have access to a smooth inductive definition of $\mathcal{L}$ and thus of the set of possible measurements according to the experiment described in Definition 2, various properties are readily proved using induction on the length of a derivation in $\mathcal{L}$.

Definition 5. *We say that a string $\sigma \in \tilde{\Sigma}^*$* **determines** *some observable $s \in \Sigma$ with value v, whenever the sequence of measurements corresponding to σ determines s as defined in Definition 2 with value v. We say that a string $\sigma \in \tilde{\Sigma}^*$ determines some context c, if σ determines each observable in c. We say that two strings $\sigma, \sigma' \in \tilde{\Sigma}^*$* **agree** *on $s \in \Sigma$, whenever either both do not determine s or both determine s with the same value.*

With this definition at hand, we explicitly re-state some observations that were already used in the proof of Theorem 1.

Lemma 1. *If some $\sigma \in \tilde{\Sigma}^*$ determines a context, then any extension/continuation $\sigma \star \tau$ of σ also defines a context.*

This lemma does not scale to systems of more qubits. The following lemma does.

Lemma 2. *Each $\sigma \in \tilde{\Sigma}^*$ determines at most one context.*

3 Hidden variables and de-randomization

In this section we wish to study the cost of various forms of de-randomization of our experiment. First, the rather evident observation is made that a finite set of hidden finite valued variables can never account for indeterminism in an ideally isolated repeatable experiment. Later we shall elaborate on some refinements of this statement.

3.1 Need for infinitely many hidden discrete variables

Let us reconsider our experiment from Section 2 where we measure our sequence of observables. There is an amount of randomness present. If we measure A for the first time, the outcome can be both 1 or -1. However, it might be the case that the particular outcome is dependent on some additional parameters that we can not directly observe. We shall refer to those additional parameters as *hidden variables* (HVs).

The question now raises, can we find a model that uses a finite amount of hidden variables such that, once they have been assigned some initial values at the outset of our experiment, then the outcome of the experiment becomes fully deterministic. We shall first see that the answer is NO if our hidden variables can only take on a finite amount of values and under some additional general assumptions. Later we shall fine-tune on this result.

Let us first formulate these two general additional assumptions that from now on shall be implicitly assumed throughout the rest of this section.

> **Assumption of independence of exact time** We assume that in our experiment, the exact outcome of our measurement does not depend on the exact time the measurement has taken place. The only important thing is the history of measurements so far. Thus, for any positive t_1 and t_2, measuring B after t_1 seconds after measuring A should yield the same value as B after t_2 seconds after measuring A, Ceteris Paribus.

Moreover, we shall assume that there is no relevant interaction between our experiment and the outside world. Thus, with these two assumptions, a deterministic explanation of our experiment will exist of a function

$$f(\boldsymbol{x}, A): \quad H \times \Sigma' \quad \to H,$$

where Σ' is the set of observables, in our case $\Sigma' = \Sigma = \{A, B, C, a, b, c, \alpha, \beta, \gamma\}$ and H is some space where the (hidden) variables attain their values. Thus basically, the function f will tell you, once you know all the underlying hidden variables in $\boldsymbol{x}$, what the next value will be. So, the idea is that the hidden variables fully determine the situation and the outcome of future measurements. Thus, for convenience, and without loss of generality we may assume that $\Sigma \subseteq H$.

We can easily extend f in such a way that given the initial situation in terms of the HVs we can obtain a full description, again in terms of the HVs, of any future situation given that we decide to perform a sequence of measurements. If σ is some sequence of observables, that is, $\sigma \in \Sigma^*$, we will denote by $f^\sigma(\boldsymbol{x})$ the values of the HVs after performing this sequence σ of measurements when we start with initial conditions $\boldsymbol{x}$. Formally, we define this as follows.

Definition 6 (f^σ). *Let f be a deterministic explanation of our experiment and let $\sigma \in (\Sigma')*$, that is, σ is a string of observables. The function f^σ is defined inductively on the length of the string σ as follows:*

$$\begin{array}{ll} f^\lambda := \mathbb{I} & \text{where } \mathbb{I} \text{ is the identity operator;} \\ f^{\tau\star A}(\boldsymbol{x}) := f(f^\tau(\boldsymbol{x}), A) & \text{where } \star \text{ is the concatenation operator.} \end{array}$$

Now basically, if $\boldsymbol{x}$ takes its values in some discrete and finite space H, then there are only finitely many starting conditions $\boldsymbol{x}$. With these finitely many degrees of freedom in the starting conditions, we can never account for the infinitude of choices that can be made in our sequence of measurements. We can make this idea more formal in the following easy theorem.

Theorem 2. *Let f be a deterministic explanation of our experiment and let H be the space values where the hidden variables take on their values. If H is finite, then it can never account for all the possible outcomes of our experiment. That is, in this case, the hidden variables do not fully determine the full course of measurements to follow.*

Proof. The basic idea is that, if H is finite, then $\boldsymbol{x}$ only can be a finite number of different initial conditions. A different sequence of measurements can only occur when the initial values of $\boldsymbol{x}$ were different. Thus, for example, the outcome $A = 1; b = 1$ must necessarily have started with a different initial condition $\boldsymbol{x}$ than the sequence $A = 1; b = -1$. As there are infinitely many different sequences that are in our language $\mathcal{L}$, they can only be accounted for by infinitely many different initial conditions.

3.2 Refinements and lower-bounds

Now that we have seen in Theorem 2 that no finite amount of HVs suffices to de-randomize our experiment, we can ask ourselves the following questions concerning refinements of Theorem 2.

1. What is the amount of binary HVs needed to explain all experiments of length n. That is, how many bits of memory are needed at least to explain from the initial conditions the full outcome of all possible sequences of measurements up to length n?
2. What is the minimal amount of binary memory needed to *recognize* any string in $\mathcal{L}$ by walking linearly through that string? Notice, in this question the aim is not to de-randomize the experiment by using HVs, rather it asks how much memory is needed in order to walk through a string of the form $A\tilde{b}Ab\gamma C\tilde{c}$ without necessarily copying the entire string in some memory, and tell in the end[4] of it whether or not the string is in $\mathcal{L}$.

Question 2 and a variation thereof are addressed in the next section. An answer to Question 1 essentially amounts to counting the number of strings up to length n and then taking the logarithm of that number. Note that, in the light of Proposition 1 and Corollary 1 we know the order of magnitude of this number. That is, we know that the order of magnitude of number strings of length n in $\mathcal{L}$ is (disregarding the polynomials, which are negligible on a logarithmic scale) λ^n, whence the order of strings up to length n is about λ^{n+1} and the logarithm of that results in a growth linear in n. Note that the total number of strings of length up to n in $\tilde{\Sigma}^*$ is of order $|\Sigma^*|^{n+1}$ so that the logarithm of that is also of order linear in n. In this sense, the number of HVs needed to predict all experiments up to length n is as bad (high) as it could possibly be.

4 Hidden variables and the Holevo bound

Holevo [6] showed that the maximum information carrying capacity of a qubit is one bit. Therefore, a machine which simulates qubits but has a density of memory (in bits per qubit) larger than one violates the Holevo bound. In this section we show that a very broad class of machines that simulate the Peres-Mermin square violate the Holevo bound.

In [7] deterministic automata are presented that generate a subsets of $\mathcal{L}$. In that paper lower bounds on the amount of states of these automata are presented. Of course, as $\mathcal{L}$ inhibits a genuine amount of non-determinism any deterministic automata will generate only a proper subset of $\mathcal{L}$ but never the whole set $\mathcal{L}$ itself.

In the next subsection we shall introduce the notion of an Memory-factoring Abstract Generating Automata (MAGA) for $\mathcal{L}$ and prove a lower bound on the number of states any MAGA should have if it were to generate $\mathcal{L}$. In a sense, a MAGA for $\mathcal{L}$ will generate all of $\mathcal{L}$.

[4] So this particular string $A\tilde{b}Ab\gamma C\tilde{c}$ is in $\mathcal{L}$.

4.1 Memory-factoring Abstract Generating Automata

In this project we are not interested in the details of (abstract) machine 'hardware'. Thus, in our definition of a MAGA we will try to abstract away from the implementation details of language generating automata. We will do this by invoking the notion of *computability*. By the Church-Turing thesis (see, a.o. [10]) any sufficiently strong and mechanizable model of computation can generate the same set of languages, or equivalently, solve the same same set of problems. Thus, instead of fixing one particular model of computation and speak of computability therein, we may just as well directly speak of computable outright leaving the exact details of the model unspecified.

Basically, a MAGA $\mathcal{M}$ for $\mathcal{L}$ is an abstract machine that will predict the outcome of a measurement of some observable $s \in \Sigma$ in the experiment as described in Definition 2 given that a sequence of measurements $\sigma \in \tilde{\Sigma}^*$ has already been done. If according to the experiment s is determined by σ with value $v \in \{1, -1\}$, then $\mathcal{M}$ should output v. If the observable s is not determined by σ then $\mathcal{M}$ should output r indicating that the experiment can randomly output a -1 or a 1.

The only requirement that we impose on a MAGA is that its calculation in a sense *factors through* a set of memory states[5] in the sense that before outputting the final value, the outcome of the calculation is in whatever way reflected in the internal memory of the machine. Let us now formulate the formal definition of a MAGA for $\mathcal{L}$.

Definition 7 (MAGA). *A* Memory-factoring Abstract Generating Automata *(MAGA) for $\mathcal{L}$ is a quadruple $\langle M, M_0, M_1, S\rangle$ with*

1. *S is (finitely or infinitely) countable set of memory states;*
2. *all of M, M_0 and M_1 are computable functions such that*
   (a) *$M = M_1 \circ M_0$;*
   (b) *$M_0 : \mathcal{L} \times \Sigma \to S \times \Sigma$,*
   where[6] *$\Pi_2 \circ M_0 = \mathbb{I}$;*
   (c) *and $M_1 : S \times \Sigma \to \{1, -1, r\}$,*

   such that

$$\begin{array}{ll} M(\sigma, s) = 1 & \text{if } \sigma \text{ determines } s \text{ with value } 1; \\ M(\sigma, s) = -1 & \text{“ ” } -1; \\ M(\sigma, s) = r & \text{if } \sigma \text{ does not determine } s. \end{array}$$

In our definition, we have that $M_0 : \tilde{\Sigma}^* \times \Sigma \to S \times \Sigma$, where M_0 does nothing at all on the second coordinate, that is, on the Σ part. We have decided to nevertheless take the second coordinate along so that we can easily compose M_0 and M_1 to obtain M. This is just a technical detail. The important issue is that the computation *factors* through the memory. That is, essentially we have that $M : \mathcal{L} \times \Sigma \xrightarrow{M_0} S \xrightarrow{M_1} \{1, -1, r\}$ where 'M_1 borrows some extra information on the Σ-part of the original input'.

Theorem 3. *The class of MAGA-computable functions is the full class of computable functions.*

Proof. It is easy to see that if we have infinite memory, we can conceive any Turing Machine μ with the required input-output specifications as a MAGA where M_0 is just the identity, S is coded by the tape input, and M_1 is the function computed by μ. Thus, the class of MAGA-computable functions is indeed the full class of computable functions.

[5] A memory state is something entirely different from, and hence is not to be confused with, a quantum state.

[6] Here Π_2 is the so-called projection function that projects on the second coordinate: $\Pi_2(\langle x, y\rangle) = y$. Basically, $\Pi_2 \circ M_0 = \mathbb{I}$ just says that M_0 only tells us which state is defined by a sequence $\sigma \in \tilde{\Sigma}^*$.

4.2 A lower bound for the Peres-Mermin square

With the formal definition at hand we can now state and proof the main theorem for the Peres-Mermin square. In the proof we will use the so-called Pigeon Hole Principle (PHP). The PHP basically says that there is no injection of a finite set into a proper subset of that set. Actually we will only use a specific case of that which can be rephrased as, if we stuck $n+1$ many pigeons in n many holes, then there will be at least one hole that contains at least two pigeons.

Theorem 4. *Any MAGA for $\mathcal{L}$ contains at least 24 states. That is, if $\langle M, M_0, M_1, S\rangle$ is such a MAGA, then $|S| \geq 24$.*

Proof. We shall actually use a slightly modified version of a MAGA to prove our theorem. In the unmodified MAGA, the function M_0 tells us what state is attained on what sequence of measurements $\sigma \in \tilde{\Sigma}^*$. In the modified MAGA we will only require that M_0 will tell us in what state the machine is whenever σ determines a full context.

To express this formally we define

$$\tilde{\Sigma}^+ \;:=\; \{\sigma \in \tilde{\Sigma}^* \mid \sigma \text{ determines a full context of observables}\}.$$

Thus, instead of requiring that M_0 maps from $\tilde{\Sigma}^* \times \Sigma$ to $S \times \Sigma$, we will require that M_0 maps from $\tilde{\Sigma}^+ \times \Sigma$ to $S \times \Sigma$. Clearly, if we have a lower bound for any MAGA $\mathcal{M}$ with this restriction on M_0, we automatically have the same lower bound for any MAGA $\mathcal{M}'$ outright. This is so as any MAGA $\mathcal{M}'$ trivially defines a restricted MAGA $\mathcal{M}$ by just restricting the domain of M_0 to $\tilde{\Sigma}^+ \times \Sigma$.

To continue our proof, let $s_1, \ldots, s_{24}$ enumerate all possible combinations $\langle c, v_1, v_2\rangle$ of contexts and the first two[7] values of the first two observables of that context c. Note that there are indeed $6 \times 2 \times 2 = 24$ many such combinations. We define a map

$$C \;:\quad \tilde{\Sigma}^+ \;\to\; \{s_1, \ldots, s_{24}\}$$

in the canonical way, mapping an element $\sigma \in \tilde{\Sigma}^+$ to that s_i that corresponds to the triple consisting of the context that is determined by σ followed by the first two values of the the first two observables of that context. By Lemmas 1 and 2 the function C is well-defined.

Now, let $\sigma_1, \ldots, \sigma_{24}$ be representatives in $\tilde{\Sigma}^+$ of $s_1, \ldots, s_{24}$ such that $C(\sigma_i) = s_i$. For a contradiction, let us assume that there exists some restricted MAGA $\langle M, M_0, M_1, S\rangle$ for $\mathcal{L}$ with $|S| \leq 23$. By the Pigeon Hole Principle, we can choose for this MAGA some σ_i and some different σ_j such that[8]

$$M_0(\sigma_i) = M_0(\sigma_j).$$

We now use the following claim that shall be proved below. Recall from Definition 5 what it means for two sequences to agree on some variable.

Claim. If $\sigma_k \neq \sigma_l$ then there is some $s \in \Sigma$ such that σ_k and σ_l disagree on s.

Once we know this claim to hold it is easy to conclude the proof. Consider any $s \in \Sigma$ on which σ_i and σ_j disagree. By the definition of M we should have that $M(\sigma_i, s) \neq M(\sigma_j, s)$. However as $M_0(\sigma_i, s) = M_0(\sigma_j, s)$ we see that $M_1 \circ M_0(\sigma_i, s) = M_1 \circ M_0(\sigma_j, s)$. But $M_1 \circ M_0 = M$, which contradicts $M(\sigma_i, s) \neq M(\sigma_j, s)$. We conclude that M can not have 23 or less states.

[7] For horizontal contexts we will enumerate from left to right and for vertical contexts we will enumerate from top to bottom. Thus, for example, the first two observables of the context $\langle C, c, \gamma\rangle$ are C and c.

[8] Par abus de langage we will write $M_0(\sigma_i)$ as short for $\Pi_2(M_0(\sigma_i, s))$.

Thus to finalize our proof we prove the claim. Let $C(\sigma_k) = \langle c^k, v_0^k, v_1^k\rangle \neq \langle c^l, v_0^l, v_1^l\rangle = C(\sigma_l)$. If $c^k \neq c^l$ then c^k contains at least two observables on which σ_k and σ_l agree as each of these σ's only determine observables in their respective contexts.

In case $c^k = c^l$, then one of v_0^k, v_1^k differs from the corresponding one in v_0^l, v_1^l giving rise to a disagreement between σ_k and σ_l.

This concludes the proof of the claim and thereby of Theorem 4.

Remark 2. Note that the proof of Theorem 4 nowhere invokes the notion of computability therefore proving actually something stronger.

One can easily see that for $\mathcal{L}^+$ the obtained lower bound is actually sharp in the sense that there is a MAGA with 24 memory states for $\mathcal{L}^+$. However, is seems that for $\mathcal{L}$ this is not the case.

Any memory-factoring device that *recognizes* $\mathcal{L}$ –let us call that a MARA for convenience– can be turned into a MAGA for $\mathcal{L}$. In analogy with our MAGA, we conceive such a device as some $M' : \mathcal{L} \times \tilde{\Sigma} \xrightarrow{M_0'} S' \times \Sigma \xrightarrow{M_1'} \{\text{YES}, \text{NO}\}$. Now, any such device can be transformed to a MAGA $\langle M, M_0, M_1, S\rangle$ with $S = S' \times S'$ and M_0 maps any pair $\langle \sigma, s\rangle$ to the pair $\langle M_0'(\sigma, s), M_0'(\sigma, \overline{s})\rangle$. Then M_1 will output r only if both σs and $\sigma\overline{s}$ are in $\mathcal{L}$ and do the obvious thing otherwise. To put it more formal, M_1 acts on a pair from $S' \times S'$ using M_1' for each coordinate and consequently mapping (YES, YES) to r, mapping (YES, NO) to 1, and[9] mapping (NO, YES) to -1. Note that if n is a lower bound for a MAGA, then the lower bound on a MARA that corresponds to this construction is of size $\sqrt{n}$ thus giving a partial answer to Question 2 from Section 3.2.

4.3 Scaling

We first note that the proof of Theorem 4 is very amenable to generalizations:

Remark 3. The proof of Theorem 4 easily generalizes under some rather weak conditions giving rise to lower bounds of #contexts $\times\, 2^{(\#\text{degrees of freedom in one context})}$.

The Peres-Mermin square (†) that corresponds to the two-qubit system can be generalized in various ways [1, 2]. One particular generalization for n qubits gives rise to a system where each context consists of exactly d elements with $d = 2^n$ [2]. Moreover, there are $c := \prod_{k=1}^{m}(2^k + 1)$ different many such contexts. Each context is now determined by a particular selection of n of its elements. We shall denote the corresponding languages by $\mathcal{L}_n$. Thus, what we have called $\mathcal{L}$ so far in this paper, would correspond to $\mathcal{L}_2$.

Theorem 5. *Any MAGA for $\mathcal{L}_n$ contains at least $2^n \cdot \prod_{k=1}^{n}(2^k + 1)$ many different memory states.*

Proof. Basically this is just by plugging in the details of the languages $\mathcal{L}_n$ into remark 3. Let us very briefly note some differences with the proof of Theorem 4. The main difference is that the set $\mathcal{L}_2^+$ is nice: once a string is in there, any extension is as well. However, this does not impose us to, again define a restricted MAGA by restricting the domain of M_0 to $\mathcal{L}_n^+$. Again, we consider the $n+1$-tuples consisting of a context with some values for the[10] n observables that determine this context and choose some correspondence between these tuples and some representing sequences $\sigma_i \in \tilde{\sigma}_n^*$. Clearly, these $\sigma_i \in \mathcal{L}_n^+$. As the claim obviously holds also in the general setting, the assumption that the memory states $s_1, \ldots, s_{2^n \cdot \prod_{k=1}^{n}(2^k+1)-1}$ suffice yields together with the PHP to a contradiction as before.

[9] Note that (NO, NO) cannot occur.

[10] For each context, we fix some n observables that determine that context as these are not uniquely defined

As was done in [3] and in [7], we can consider the information density d_n for the corresponding languages defined as the number of classical bits of memory needed to simulate a qubit:

$$d_n := \frac{\log_2(|S_n|)}{n}. \qquad (+)$$

If we apply the lower bound for S_n –the number of memory states for a MAGA for $\mathcal{L}_n$– from Theorem 5 to (+) we obtain

$$\begin{array}{rcl} S_n & \geq & 2^n \cdot \prod_{k=1}^{n}(2^k+1) \\ & \geq & 2^n \cdot \prod_{k=1}^{n}(2^k) \\ & \geq & 2^n \cdot (2^{\sum_{k=1}^{n} k}) \\ & \geq & 2^n \cdot (2^{\frac{n(n+1)}{2}}) \end{array}$$

whence d_n is approximated (from below) in the limit by $\frac{\log_2(2^n \cdot (2^{\frac{n(n+1)}{2}}))}{n} = \frac{n+3}{2} \sim \frac{n}{2}$. Thus, the information density in this generalization of the Peres-Mermin square grows linear in the number of qubits. However, as observed before, any density more than 1 implies a violation of the Holevo bound.

References

1. A. Cabello, Proposed test of macroscopic quantum contextuality, Phys. Rev. A **82**, 032110 (2010).
2. A. Cabello *et al.*, State-independent quantum contextuality for n qubits, arXiv:1102...
3. A. Cabello, The role of bounded memory in the foundations of quantum mechanics. Found. Phys., published online 16 September 2010. DOI: 10.1007/s10701-010-9507-2.
4. N. Chomsky, Three models for the description of language. IRE Transactions on Information Theory (2): 113124 (1956).
5. P. Flajolet and R. Sedgewick, Analytic Combinatorics, Cambridge University Press, New York, ISBN 978-0-521-89806-5 (2009).
6. A. S. Holevo, Some estimates of the information transmitted by a quantum communication channel. Probl. Inf. Trans., (9); 177 (1973).
7. M. Kleinmann, O. Gühne, J.R. Portillo, J-Å. Larsson, and A. Cabello, Memory cost of quantum contextuality. arXiv:1007.3650.
8. N. D. Mermin, Simple unified form for the major no-hidden-variables theorems. Phys. Rev. Lett. **65**, 3373 (1990).
9. A. Peres, Incompatible results of quantum measurements. Phys. Lett. A **151**, 170 (1990).
10. M. Sipser, Introduction to the Theory of Computation. PWS Publishing. ISBN 0-534-94728-X (1997).