INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Exhaustive enumeration of Kochen-Specker vector systems*

M. Pavičić — J-P. Merlet — N.D. Megill

**N° 5388**

Novembre 2004

Thème SYM


ISSN 0249-6399 ISRN INRIA/RR--5388--FR+ENG

INRIA
SOPHIA ANTIPOLIS

# Exhaustive enumeration of Kochen-Specker vector systems

M. Pavičić* , J-P. Merlet† , N.D. Megill ‡

Thème SYM — Systèmes symboliques
Projet Coprin

Rapport de recherche n° 5388 — Novembre 2004 — 39 pages

**Abstract:** The Kochen-Specker theorem is one of the fundamental theorems in quantum mechanics. It proves that the *non-contextuality* hypothesis that assumes that the values of observables are independent of the context, i.e., of the way that measurements are performed on the system and therefore predetermined, does not hold for quantum systems. The proof is provided by any counterexample to the assumed pre-existence of values of *observables* in quantum mechanics (such as the spin of a system). The theorem not only characterizes quantum systems but is also one of the major arguments against *hidden variables* theories that assume that the ambiguity of the measurements of observables may be ascribed to hidden variables that are not measured.

We give a constructive and exhaustive definition of Kochen-Specker (KS) vectors in a Hilbert space of any dimension as well as of all the remaining vectors of the space. KS vectors are elements of any set of orthonormal states, i.e., vectors in $n$-dim Hilbert space, $\mathcal{H}^n$, $n \geq 3$ to which it is impossible to assign 1s and 0s in such a way that no two mutually orthogonal vectors from the set are both assigned 1 and that not all mutually orthogonal vectors are assigned 0. Our constructive definition of such KS vectors is based on algorithms that generate linear MMP diagrams corresponding to blocks of orthogonal vectors in $\mathbb{R}^n$, on algorithms that single out those diagrams on which algebraic 0-1 states cannot be defined, and on algorithms that solve nonlinear equations describing the orthogonalities of the vectors by means of interval analysis. To demonstrate the power of the algorithms, all 4-dim KS vector systems containing up to 24 vectors were generated and described, all 3-dim vector systems containing up to 30 vectors were scanned, and several general properties of KS vectors are found.

**Key-words:** quantum mechanic , symbolic computation, systems solving, interval analysis

* University of Zagreb, Gradjevinski fakultet, POB 217, HR-10001 Zagreb, Croatia, E-mail: pavicic@grad.hr
† INRIA Sophia-Antipolis, E-mail: Jean-Pierre.Merlet@sophia.inria.fr
‡ Boston Information Group, 30 Church St., Belmont MA 02478, U. S. A., E-mail: nm@alum.mit.edu

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

# Enumération exhaustive des systèmes de vecteurs de Kochen-Specker

**Résumé :** Le théorème de Kochen-Specker est un des théorème fondamentaux de la mécanique quantique. Il montre que l'on ne peut pas supposer que les valeurs des observations sont indépendantes du contexte, c'est-à-dire de la manière dont sont mesurées les observations. La preuve en repose sur l'existence de contre-exemples qui montre qu'il n'est pas possible d'assigner des valeurs déterminées aux observations d'un système quantique (par exemple le spin d'un système). Ce théorème est un des arguments majeurs de réfutation de la théorie des *variables cachées*, qui suppose que l'ambiguïté des mesures sur un système quantique est attribuable à l'existence de variables cachées qui ne sont pas mesurées.

Nous proposons un processus constructif qui permet une énumération exhaustive des contre-exemples. Pour cela nous définissons les systèmes de vecteurs de Kochen-Specker comme un ensemble de vecteurs d'un espace de Hilbert $\mathcal{H}^n$ de dimension $n$, $n \geq 3$, pour lequel il est impossible d'assigner un état 0 ou 1 à chaque vecteur de façon à ce que les vecteurs de toute paire de vecteurs mutuellement orthogonaux n'aient pas simultanément un état 1 et que tous les vecteurs appartenant à un ensemble de vecteurs mutuellement orthogonaux n'aient pas simultanément un état 0.

Le processus constructif repose sur une énumération exhaustive de toutes les séquences constituées de $n$ groupes de vecteurs parmi un ensemble donné de $a$ vecteurs. Dans cette énumération on retient ensuite seulement les séquences pour lesquelles on ne peut pas assigner un état 0 ou 1 aux vecteurs constitutif de la séquence. Les séquences obtenues constitueront alors un système de vecteurs de Kochen-Specker s'il est possible d'attribuer aux vecteurs des composantes réelles. Examiner cette possibilité revient à résoudre le système d'équations résultant des conditions d'orthogonalité entre les vecteurs, dont les inconnues sont les composantes des vecteurs. Pour cela une approche reposant sur la propagation de contraintes et l'analyse par intervalles est utilisée.

La puissance de ce processus est illustrée par la génération de tous les systèmes de vecteurs de Kochen-Specker en dimension 4 contenant jusqu'à 24 vecteurs ainsi que ceux contenant jusqu'à 30 vecteurs en dimension 3. On en déduit plusieurs propriétés générales des systèmes de vecteurs de Kochen-Specker.

**Mots-clés :** mécanique quantique, calcul formel, résolution de systèmes, analyse par intervalles

# 1 Introduction

The Kochen-Specker theorem is one of the fundamental theorems in quantum mechanics. It proves that the *non-contextuality* hypothesis does not hold for quantum systems. This hypothesis assumes that the values of observables are independent of the context, i.e., of the way that measurements are performed on the system and therefore predetermined, what holds for observables of classical systems. The proof is provided by any counterexample to the assumed pre-existence of values of *observables* in quantum mechanics (such as the spin of a system). The theorem not only characterizes quantum systems but is also one of the major arguments against *hidden variables* theories that assume that the ambiguity of the measurements of observables may be ascribed to hidden variables that are not measured.

Recently proposed experimental tests of the Kochen-Specker (KS) theorem [7, 28], disputes on feasibility of such experiments [18, 12, 17, 27, 4] and its first experimental verification [9] prompted a renewed interest in the theorem.

To prove the Kochen-Specker theorem, one chooses projectors $P_i$, $i = 1, \ldots, n$ to vectors (orthonormal states) in $n$-dim Hilbert space, $\mathcal{H}^n$. For them Kochen and Specker [15] showed that there is no function $f : \mathcal{H} \to \mathbb{R}$ satisfying the Sum Rule: $\sum_{i=1}^{n} f(P_i) = f(\sum_{i=1}^{n} P_i) = f(I)$ for all sets of projectors $P_i$. If one chooses $f(P_i) \in \{0, 1\}$ ($f(I) = 1$) it follows that it is impossible to assign 1s and 0s to all vectors in such a way that (1) no two mutually orthogonal vectors are both assigned 1; (2) mutually orthogonal vectors cannot all be assigned 0.

We recognize that a description of a discrete observable measurement (e.g., spin) in $\mathcal{H}^n$ can be rendered as a `0-1` measurement of the corresponding projectors along orthogonal vectors in $\mathbb{R}^n$ to which the projectors project. Every set of such vectors that satisfy the Kochen-Specker in the sense of violating the Sum Rule we call a *Kochen-Specker set* and their elements *Kochen-Specker vectors*. Kochen-Specker vectors correspond to experiments having no classical counterparts. In our previous paper [23] we determined the class of all Kochen-Specker vectors using several algorithms following the ideas put forward in [20, 19, 21]. Our constructive definition of such KS vectors is based on the algorithms that generate linear MMP diagrams corresponding to blocks of orthogonal vectors in $\mathbb{R}^n$, on algorithms that filter out diagrams on which algebraic `0-1` states cannot be defined, and on algorithms that solve nonlinear equations describing the orthogonality of the vectors by means of polynomially complex interval analysis and self-teaching programs. In this paper we present the details of the algorithms with special attentions to methods of solving nonlinear equations obtained by means of the algorithms.

We shall denote Kochen-Specker vectors by `1,2,...,9,A,B,...Z,a` ... and their number within a Kochen-Specker set in $\mathbb{R}$ by $a$. It is necessary to assume that all the vectors in this set are *independent* i.e. no pair of vectors in the set should be identical or opposite, otherwise if we have $m$ collinear vectors we will measure the spin along only $a - m + 1$ directions. This constraint will be called the *non-collinearity constraint*.

All vectors in the set will be mutually orthogonal with at least $n - 1$ other vectors in the set while a given vector may be mutually orthogonal with more than one family of $n - 1$ vectors. These vectors will be assembled in a system of $b$ groups of $n$ elements such that all the vectors in a group are mutually orthogonal (this will be called the *orthogonality*

*constraint*) and the system may be described using the following notation: each group is written in turn, each group being separated by comma as for example:

`1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH`

which denotes a system with $a = 18, b = 9$. Note in this example that several vectors appear in different groups and that a description of a given measurements system is not unique as an exchange in the lettering will lead to the same measurements system. Systems that correspond to the same measurement arrangements will be called *isomorphic systems*.

This system may be represented graphically as a diagram with vertices representing vectors and edges representing groups of vectors. The *loop* size of a diagram is the number of edges of the minimal polygon that may be found in the diagram.

The components of a vector in such system will be denoted $aij$ with $i$ in $[1, a]$ and $j$ in $[1, n]$.

Since a vector in a group represents a measurement axis for a quantum system, a measurement gives either result 1 (a "click") or 0 (no "click"). In a given group, we get 0 along $n - 1$ vectors and 1 along the remaining one. However, we cannot assume that the system possessed these values for the measured observable prior to its measurement and this means that we get the Kochen-Specker vectors—as examples of this no-go feature of quantum systems—as a set of vectors to which it is impossible to assign `0-1` states in the aforementioned way.

Let us first see what constraints should be imposed upon vectors before we try to verify whether they are Kochen-Specker vectors or not:

- all vector components should be real (for measurement purposes we are interested only in real Kochen-Specker systems)
- all vectors in a group are mutually orthogonal
- no vector in the solution set should be collinear with another vector in the solution set. This constraint is imposed by the measurement devices that cannot measure exactly in the same direction. Hence it may be somewhat relaxed as we will see later on.

Investigating Kochen-Specker systems is important from two view points:

- for a given $n$ obviously the smaller $a$ is, the easier it is to carry out the experiment. Even more so as recently a single qubit KS scheme was formulated [5] by means of auxiliary quantum systems (ancillas) of the measuring apparatus and subsequently connected with the original KS formulation [1].
- experiments may involve various $n$ and we must be able to exhibit Kochen-Specker vectors for any $n$

There are known Kochen-Specker vector systems that have solution and subsequent attempts to reduce the number of vectors are usually called *records*.

For example the system with $a = 18, b = 9$:

`1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH`

known as Cabello's system [6] with loop of size 3 has a solution (see figure 1 for a graphical representation of this system).

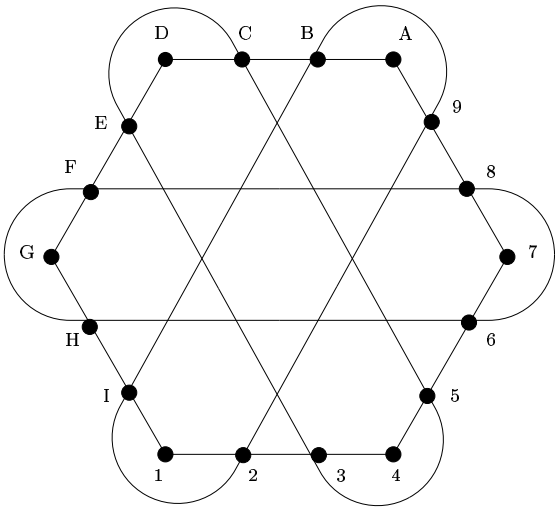



Figure 1: Cabello's system, the smallest 4-dim Kochen-Specker real system ($a = 18, b = 9$)

Another system with solution is:

`1234,1256,1378,149A,BCDE,BDF8,CD9G,DE5H,AGIJ,6HIK,F7KJ`

known as the Kernaghan system [13] with 20 vectors and loops of size 2.

In dimension 3 known systems are the Peres system [25]:

```
123,39R,89A,47D,56E,DRE,EFG,CBD,NML,LKE,DJQ,QST,PJI,HKO,RVX,RUW,14Y,1Z5,
4aA,5b8,8gB,AhF,7cH,6dI,CiO,GjP,7eM,6fS,ClN,GkT,NqX,PsV,OrU,MmU,SnV,HoX,
IpW,TtW,2uB,2vF
```

with 57 vectors while Conway-Kochen's has 51 vectors [26, p. 114]. Another system with 49 vectors is Bub's system [2]:

```
123,345,167,AB6,AC4,DEG,DFH,F9O,E8V,5JI,7MN,GIa,HNh,7LT,5KR,DAe,UTS,
PRS,1GP,3HU,3Vj,Pgh,Uba,1Oi,VZg,OYb,6Xk,4Wn,Sde,dci,dfj,imn,jlk,akQ,hnQ,eQ2
```

The reasons for referring to these systems as 57, 51, and 49, and not 33, 31, and 33 vector systems are given in [21].

There are several solutions for larger spaces. One of them is Kernaghan's solution in 8 dimensional space [14].

Prior to our approach [20, 19, 21] no general method for constructing sets of KS vectors was known and the KS vectors were constructed either by means of partial Boolean algebras and orthomodular lattices [15, 29, 30, 31], by direct experimental proposals [7, 3, 28], by combining rays in $\mathbb{R}^n$ [25, 13, 2] or by geometric intuition [24]. These approaches had two disadvantages: first, they depended on human ingenuity to find ever new examples and "records," and secondly, their complexity grew exponentially with increasing number of dimensions and vectors. E.g., lattices of orthogonal $n$-tuples have $2^n$ elements (Hasse diagrams) [10] and, on the other hand, the complexity of nonlinear equations describing combinations of orthogonality also grows exponentially.

Our purpose was to develop a method that will be able to exhibit all Kochen-Specker systems in arbitrary dimensions with a reasonable number of vectors.

This problem issues from a contact of one of us (M. Pavičić) with a former member of the Galaad project, Ioanis Emiris. M. Pavičić was looking for algorithms that would enable a realistic implementation of his constructive definition of arbitrary Kochen-Specker vectors he put forward [19] combining *McKay-Megill-Pavičić diagrams* (MMP), states defined on them [22], and nonlinear equations of the inner products they should correspond to. I. Emiris was consulted as a well-known specialist of algebraic sparse systems that appear when looking at the orthogonality constraints. Emiris (and other specialists such as D. Manocha) did not believe that sparse resultant or other algebraic geometry methods such as Gröbner basis were powerful enough to be able to deal with the large number of equations of such systems (without mentioning the difficulty of dealing with the non-collinearity constraint). I. Emiris mentioned that an alternate approach was to use interval analysis and advised M. Pavičić to contact the COPRIN project.

# 2 Generating Kochen-Specker systems

Our approach is to determine all possible Kochen-Specker systems for given $a, b, n$. For this purpose we will proceed along three steps:

1. determine an exhaustive list of MMP diagrams that are not isomorphic,
2. among these systems determine the ones for which we cannot assign dispersion free `0-1` states,
3. determine among the remaining systems the ones satisfying the orthogonality and non-collinearity constraints.

## 2.1 Generation of non-isomorphic MMP diagrams

We start with describing vectors as vertices (points) and orthogonality between them as edges (lines connecting vertices) thus obtaining MMP diagrams [19, 16, 21], which are defined as follows:

1. Every vertex belongs to at least one edge;

2. Every edge contains at least 3 vertices;
3. At least one vertex within each edge must share two edges;
4. Edges which intersect each other in $n-2$ vertices contain at least $n$ vertices

Powerful software developed by Brendan D. McKay [16] in cooperation with M. Pavičić and N. D. Megill serves us to produce all MMP diagrams for given $a, b, n$, filtering out the isomorphic one. For given $a, n$ this software produces systems incrementally by adding groups so that the non-isomorphic condition is satisfied.

But the number of generated systems is usually extremely large: for example for $a = 10, b = 12$ there are 197 885 058 systems. This number grows exponentially with $a$. For $a = 11, b = 12$ we stopped the calculation after ten hours although the generation was not completed and over 630 000 000 systems were already generated.

## 2.2 Filtering out MMP diagrams admitting 0-1 states

The only MMP diagrams of interest are those that cannot be assigned a 0-1 (dispersion free) state. The next step after generation of non-isomorphic MMP diagrams is to process them with a filter program called `states01.c`, which was written by N. Megill. This program efficiently identifies whether a 0-1 state can be assigned to an MMP diagram.

The criterion for assigning 0-1 states is that each group must contain exactly one vector assigned to 1, with the others assigned to 0. As soon as a vector in a group is assigned a 1, all other vectors in that group become constrained to 0, and so on. The algorithm in `states01.c` performs exhaustive assignment attempts to an MMP diagram, backtracking when there is a conflict.

The algorithm scans the groups then the vectors within each group in some order, trying 0 then 1, skipping vectors constrained by an earlier assignment. When no assignment becomes possible, the algorithm backtracks and tries the next possible assignment until either all possible assignments are exhausted (no solution) or a valid assignment is found.

We will illustrate several examples of the algorithm. The first line of each example shows the MMP diagram, and the lines under it show successive iterations of the group-by-group assignment attempts. A "?" means that a vector has no state assignment yet. When a 0 or 1 state is tentatively assigned to a vector, all instances of that vector are populated with the assignment.

The first example is Cabello's MMP diagram, which has no 0-1 state. For brevity we show only the first few and last few iterations.

```
   1234,1256,1378,149A,BCDE,BDF8,CD9G,DE5H,AGIJ,6HIK,F7KJ
 1 1000,10??,10??,10??,????,????,????,????,????,????,????
 2 1000,1000,10??,10??,????,????,????,??0?,????,0???,????
 3 1000,1000,1000,10??,????,???0,????,??0?,????,0???,?0??
 4 1000,1000,1000,1000,????,???0,??0?,??0?,0???,0???,?0??
 5 1000,1000,1000,1000,1000,10?0,000?,000?,0???,0???,?0??
                           ...
265 0001,0001,0001,0100,0100,00?1,100?,000?,0???,1???,?0??
```

```
266 0001,0001,0001,0100,????,???1,??0?,??0?,0???,1???,?0??
267 0001,0001,0001,0100,0001,00?1,000?,010?,0???,1???,?0??
268 0001,0001,0001,0100,0001,0001,000?,010?,0???,1???,00??
269 0001,0001,0001,0100,0001,0001,0001,010?,01??,1???,00??
270 0001,0001,0001,0100,0001,0001,0001,0100,01??,10??,00??
271 0001,0001,0001,0100,0001,0001,0001,0100,0100,100?,00?0
272 0001,0001,0001,0100,0001,0001,0001,0100,01??,10??,00??
273 0001,0001,0001,0100,0001,0001,0001,010?,01??,1???,00??
274 0001,0001,0001,0100,0001,0001,000?,010?,0???,1???,00??
275 0001,0001,0001,0100,0001,00?1,000?,010?,0???,1???,?0??
276 0001,0001,0001,0100,????,???1,??0?,??0?,0???,1???,?0??
277 0001,0001,0001,01??,????,???1,????,??0?,????,1???,?0??
278 0001,0001,00??,01??,????,????,????,??0?,????,1???,????
279 0001,00??,00??,01??,????,????,????,????,????,????,????
280 5388,????,????,????,????,????,????,????,????,????,????
```

At iteration 266 a conflict occurs, and the algorithm backtracks. After partially recovering, the algorithm starts to backtrack again at iteration 272, finally backtracking all the way to the beginning and exhausting all possible assignments. Of the 280 iterations, 141 are backtracks.

The next example shows the same MMP diagram with the second group removed. Again we omit some of the iterations for brevity. This MMP diagram can be assigned a 0-1 state as is shown by the last iteration.

```
   1256,1378,149A,BCDE,BDF8,CD9G,DE5H,AGIJ,6HIK,F7KJ
 1 1000,1???,1???,????,????,????,??0?,????,0???,????
 2 1000,1000,1???,????,???0,????,??0?,????,0???,?0??
 3 1000,1000,1000,????,???0,??0?,??0?,0???,0???,?0??
 4 1000,1000,1000,1000,10?0,000?,000?,0???,0???,?0??
                          ...
89 0100,0100,0100,????,???0,??0?,??0?,0???,0???,?0??
90 0100,0100,0???,????,???0,????,??0?,????,0???,?0??
91 0100,0100,0010,????,???0,??1?,??0?,0???,0???,?0??
92 0100,0100,0010,1000,10?0,001?,000?,0???,0???,?0??
93 0100,0100,0010,1000,1000,001?,000?,0???,0???,00??
94 0100,0100,0010,1000,1000,0010,000?,00??,0???,00??
95 0100,0100,0010,1000,1000,0010,0001,00??,01??,00??
96 0100,0100,0010,1000,1000,0010,0001,0001,010?,00?1
97 0100,0100,0010,1000,1000,0010,0001,0001,0100,0001
98 0100,0100,0010,1000,1000,0010,0001,0001,0100,0001
```

The algorithm iterated 98 times in this case, including 44 backtracks. Recovery from the last backtrack is shown starting at iteration 91, and the last iteration shows a successful 0-1 state assignment.

An optional feature of the program, called *cluster presorting*, attempts to speed up the state assignment search. A sorted list of groups is constructed as follows. A group chosen that has the most vectors in common with the remaining groups, and it is placed at the beginning of the list. After a group is placed on the list, the remaining groups that are not yet on the list are scanned, and for each such group we compute the number of vectors it has in common with the groups already in the list. One with the most vectors in common is selected for placement at the end of the list.

When the main algorithm runs, it scans the groups in the order that they appear on this list.. Thus it will process next the group that has the most vectors in common with the groups already processed. The hope is that by processing the most severe constraints first, conflicting assignments that lead to backtracks will be resolved early.

In most cases, the cluster presorting algorithm reduces the number of iterations. The previous MMP diagram provides an example of best-case behavior:

```
   1256,1378,149A,BCDE,BDF8,CD9G,DE5H,AGIJ,6HIK,F7KJ
 1 5388,????,????,1000,10??,00??,00??,????,????,????
 2 5388,???0,????,1000,1000,00??,00??,????,????,0???
 3 5388,???0,??1?,1000,1000,0010,00??,?0??,????,0???
 4 ??1?,???0,??1?,1000,1000,0010,0010,?0??,?0??,0???
 5 ??1?,???0,??10,1000,1000,0010,0010,0010,?01?,0??0
 6 ??10,???0,??10,1000,1000,0010,0010,0010,0010,0?00
 7 ??10,??10,??10,1000,1000,0010,0010,0010,0010,0100
 8 0010,0?10,0?10,1000,1000,0010,0010,0010,0010,0100
 9 0010,0010,0?10,1000,1000,0010,0010,0010,0010,0100
10 0010,0010,0010,1000,1000,0010,0010,0010,0010,0100
```

Cluster presorting reduced the number of iterations from 98 to 10 for this example, with no backtracking at all. In other cases the improvement is less dramatic. In the example of Cabello's MMP diagram above, the number of iterations was reduced from 280 to 262, with 132 backtracks.

Cluster presorting is estimated to have approximately $O(n^2)$ behavior, whereas the main algorithm can in principle have exponential behavior.

In practice, for the diagrams of interest to us, the overhead of the cluster presorting algorithm tends to be, on average, slightly greater than the savings afforded for the main algorithm. Because most diagrams of interest will have very tight coupling throughout (i.e. many groups will have vectors in common with other groups), the main algorithm tends to be extremely fast already. Most of our runs were done without cluster presorting enabled.

In the future, if diagrams with sparser coupling are studied, cluster presorting may become more useful. For such diagrams there is a greater likelihood that exponential behavior will start to be exhibited. As a simple example, we inserted a chain of new groups in the middle of Cabello's MMP diagram as follows:

```
1234,1256,1378,149A,LMNO,OPQR,RSTU,UVWX,XYZa,abcd,BCDE,BDF8,CD9G,DE5H,
AGIJ,6HIK,F7KJ
```

This diagram clearly illustrates the exponential behavior of the backtracking algorithm. Without cluster presorting, 87 996 iterations were required to determine that the diagram admits no 0-1 state. Cluster presorting, by scanning the groups in an order that forced conflicts and backtracking to occur early in the scan, reduced the number of iterations to just 262 (which may be the minimum possible or close to it, since it is the same number needed for the original Cabello's MMP diagram as mentioned above). In general we believe that cluster presorting can help attenuate possible exponential behavior in this way.

Currently, the `states01.c` program is run separately and independently from the generation of non-isomorphic MMP diagrams. Any MMP diagram that extends an MMP diagram not admitting a 0-1 state will itself not admit a 0-1 state. If we are interested only in the smallest diagrams not admitting a 0-1 state, then it may be possible to exploit this characteristic by checking whether 0-1 state assignment is possible during the generation. However, this possibility has not yet been explored.

### 2.3 Dealing with the orthogonality and non-collinearity constraints

Our objective is to develop different algorithms that deal with the orthogonality and non-collinearity constraints. The purpose of these algorithms will be twofold

1. to take into account the orthogonality and non-collinearity constraints to reduce the complexity of the generation
2. to determine if there are components for the vectors of a given system such that the vectors satisfy the orthogonality and non-collinearity constraints. The result must be *guaranteed* (i.e. there are indeed components for the vectors such that the constraints are exactly satisfied) and the components of the vectors can be computed exactly (i.e. their values may be given as a symbolic formula such as $\sqrt{2}/2$ or with an arbitrary number of digits)

Hence the problem we are interested in is:
*Being given MMP diagrams with no 0-1 states, determine if there are vectors for such systems that satisfy the orthogonality and non-collinearity constraints.*

## 3 Conditioning Kochen-Specker systems

Without any additional assumptions any Kochen-Specker system that has one solution will have in fact an infinite number of solutions. Indeed

1. if $\mathbf{X}$ is a solution vector then $\lambda\mathbf{X}$ where $\lambda$ is an arbitrary non zero scalar, is also clearly a solution called a *multiplicative solution*
2. let $R$ be a rotation or symmetry matrix (hence a matrix that satisfies $RR^T = I^n$ with $I^n$ the identity matrix in dimension $n$). If $\mathbf{W} = \{\mathbf{W_1}, \ldots, \mathbf{W_m}\}$ is a set of solution

vectors then $R\mathbf{W}$ is also a solution vector called a *rotated solution.* Indeed assume that $\mathbf{W_i}$ must be orthogonal to $\mathbf{W_j}$ i.e. $\mathbf{W_j}^T.\mathbf{W_i} = 0$. Then we have

$$(R\mathbf{W_j})^T.R\mathbf{W_i} = \mathbf{W_j}R^TR\mathbf{W_i} = \mathbf{W_j}^T.\mathbf{W_i}^T = 0$$

As a Kochen-Specker system is defined only by orthogonality equations this shows that $R\mathbf{W}$ will also be a solution. The non-collinearity constraints will be satisfied as well.

The purpose of conditioning a Kochen-Specker system is to impose constraints on the vector components so that there will be a finite number of solutions to the system.

For that purpose we will first impose that the vectors should be unitary i.e. that the Euclidean norm of the vectors is 1. Hence if $\mathbf{W_i}$ is a solution vector only $-\mathbf{W_i}$ will be another multiplicative solution. In a later section we will see that we will impose an additional constraint so that in general $-\mathbf{W_i}$ will be excluded from the solution set. Note that as the vectors are unitary their components will always lie in the range [-1,1].

Imposing the vectors to be unitary is not sufficient to have a finite set of solution vectors, since we still have the rotated solutions. To eliminate the rotated solutions we will impose that the vectors of a specific group will be the unitary orthogonal basis of $R^n$ i.e that the $n$ vectors $\mathbf{W_i}, i \in [1, n]$ in this group have as components $aW_ij = 1$ if $i = j$, 0 otherwise. This specific group will be called the *basis group* of the system.

Hence a Kochen-Specker system will have $(a - n)n$ unknowns. As a group induces $n(n-1)/2$ orthogonality equations a Kochen-Specker system will have $(b-1)n(n-1)/2$ equations to which must be added $a-n$ equations that indicate that the vectors are unitary. If $(b-1)n(n-1)/2+(a-n)$ is greater or smaller than $n(a-n)$ we will have an over-constrained or under-constrained system. For example, the system

`1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH`

with $a = 18, b = 9, n = 4$ has $(b-1)n(n-1)/2 + (a-n) = 62$, $n(a-n) = 56$ will be an over-constrained system (and one of our results presented in this report is that there are no under-constrained MMP systems).

Note that some equations of the system will be quite simple as soon as one vector of the basis group appears in groups different from the basis group. Indeed if $\mathbf{W_i}$ is a member of the basis group with only its $i$-th component being not zero, then an orthogonality condition between $\mathbf{W_i}$ and a vector $\mathbf{Y}$ implies that the $i$-th component of $\mathbf{Y}$ is 0.

In the above example, if the basis group is chosen as `1234` the system will have 56 unknowns and 62 equations, of which 12 state that a component of one vector is 0. Hence we will end up with a system of 50 equations in 44 unknowns.

# 4 Solving Kochen-Specker systems with interval analysis

Kochen-Specker systems usually have a large number of equations. Although they are algebraic and sparse, classical algebraic solving methods such as homotopy, elimination or

Gröbner basis will have difficulties solving them as these methods are very sensitive to the number of equations in a system (and these methods are not very appropriate to deal with the non-collinearity constraint). But we have seen that all the unknowns of the equations have a value that should lie in the range [-1,1]. Hence this allows the use of solvers based on *interval analysis*.

## 4.1 Interval arithmetic

Let $f$ be a function of $l$ variables $\{x_1, \ldots, x_l\}$, and let us assume that each variable $x_i$ is constrained to lie in a given range $[\underline{x_i}, \overline{x_i}]$. In interval arithmetic the width of an interval $[\underline{x}, \overline{x}]$ is defined as $\overline{x} - \underline{x}$ and the mid-point of this interval is $(\overline{x} + \underline{x})/2$.

An *interval evaluation* of $f$ is a range $[\underline{F}, \overline{F}]$ such that $f(x_1, \ldots, x_l)$ for all $x_i \in [\underline{x_i}, \overline{x_i}]$ satisfy

$$\underline{F} \leq f(x_1, \ldots, x_l) \leq \overline{F}$$

In other words $\underline{F}, \overline{F}$ are lower and upper bounds for $f$ when the variables are constrained to lie within their ranges.

A first interesting property of an interval evaluation is that if $0 \notin [\underline{F}, \overline{F}]$, then there does not exist a set of values for the variable in their range that will cancel $f$.

There are numerous ways to obtain an interval evaluation of a given function. The easiest one is the *natural evaluation*, where each mathematical operator in the function is substituted by an equivalent operator that can be used with intervals and that satisfies the inclusion rule. For example, an interval equivalent of the $+$ operator is defined as

$$[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

The interval "+" operator satisfies the inclusion rule i.e. $\forall a \in [\underline{x}, \overline{x}], \forall b \in [\underline{y}, \overline{y}]\ \underline{x} + \underline{y} \leq a + b \leq \overline{x} + \overline{y}$.

An interesting property of interval arithmetic is that it can be implemented in such way that the interval evaluation will always include the exact result of a function value. For example, consider the simple function $f(x) = x^3 * (1/3)^3$ that must be evaluated at $x = 3$. Using `double` arithmetic the result will be 0.99999 while using interval arithmetic we will get the interval [0.9999...,1.0000...].

There are numerous available packages for interval arithmetic. We used the `BIAS/Profil` [1] package.

A drawback of the natural interval evaluation is that in general it overestimates the lower and upper bound for a function. The most curious example of this overestimation is the interval evaluation of the function $x - x$ for example for the interval [-1,1]. We get $[-1, 1] - [-1, 1] = [-2, 2]$. The reason of this overestimation is that the natural interval evaluation does not take into account the dependency between the variables. The natural interval evaluation of $x - x$ is in fact the natural interval evaluation of $x - y$ in which $x, y$ are considered as independent variables that happen to have the same range.

[1] http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html

It must be noted, however, that the size of the overestimation decreases with the width of the ranges for the variables. Furthermore, an important rule of interval arithmetic is the following:

If there is only one occurrence of each variable in a function then the interval evaluation will be exact up to round-off error, i.e. if the interval evaluation of $f(x_1, \ldots, x_l)$ is $[\underline{F}, \overline{F}]$, then there are values $x_1^m, \ldots, x_l^m$ and $x_1^M, \ldots, x_l^M$ of the variables in their range such that $f(x_1^m, \ldots, x_l^m) = \underline{F}$ and $f(x_1^M, \ldots, x_l^M) = \overline{F}$

It must be noted that the equations in a Kochen-Specker system satisfy this rule.

## 4.2 Solver based on interval analysis

We consider a Kochen-Specker system with $l$ equations in $m$ unknowns:

$$f_i(x_1, \ldots, x_m) = 0, \quad i \in [1, l]$$

A *box* is a set of ranges, one for each of the variables. The solving algorithm will use a list of $N$ boxes $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$. An interval vector of dimension $i$ $F(\mathcal{I}_j)$ will represent the interval evaluation of the $f$ functions for the box $\mathcal{I}_j$. The width of a box will be defined as the maximal width of the ranges in the box.

The proposed algorithm will return as a potential solution of a Kochen-Specker system a box $\mathcal{I}_k$ whose width will be smaller than a given threshold $\epsilon$, such that the width of $F(\mathcal{I}_k)$ will be smaller than another threshold $\epsilon_F$, each interval of $F(\mathcal{I}_k)$ including 0. The solving algorithm will look for a solution for the variables within a search space: in our case this search space is a box $\mathcal{B}$ whose ranges are all [-1,1]. When starting the algorithm the list $\mathcal{I}$ has only one element, which is the box $\mathcal{B}$.

A procedure called a *bisection* will be used on a box $\mathcal{I}_k$. In this procedure the range $I = [\underline{x_r}, \overline{x_r}]$ of one variable $x_r$ will be split in 2 intervals $I_1 = [\underline{x_r}, Mid(I)]$ and $I_2 = [Mid(I), \overline{x_r}]$ where $M(I)$ denotes the mid-point of $I$. Then we will create two boxes that are copies of $\mathcal{I}_k$ except for the variable $r$: one box will have $I_1$ are interval for $x_r$ while the other one will have $I_2$. Another procedure, the filtering of a box, will be used and will be explained in the next section.

An index $i$ is used in the algorithm and its initial value is 1.

1. if $i = N + 1$ return **NO SOLUTION**
2. filter $\mathcal{I}_i$
3. compute $F(\mathcal{I}_i)$
4. if there exist $F_k$ with $k$ in $[1, l]$ such that $\overline{F_k(\mathcal{I}_i)} < 0$ or $\underline{F_k(\mathcal{I}_i)} > 0$, then $i = i + 1$ and go to step 1
5. if the width of $\mathcal{I}_i$ is smaller than $\epsilon$ and the width of $F(\mathcal{I}_i)$ is smaller than $\epsilon_F$ then return **SOLUTION** $\mathcal{I}_i$

6. bisect $\mathcal{I}_i$ to get two new boxes $\mathcal{B}_1$ and $\mathcal{B}_2$. Substitute $\mathcal{I}_i$ by $\mathcal{B}_1$, shift the boxes at position $i+1,\ldots,N$ to position $i+2,\ldots,N+1$, store $\mathcal{B}_2$ at position $i+1$. Set $N$ to $N+1$ and go to step 1

### 4.2.1 Filtering the boxes

The purpose of the filtering is to decrease the width of a box by excluding parts of it that cannot include a solution. The filtering may even determine that a box cannot include a solution.

Two approaches will be used for the filtering step. The first one is called the 2B method [8] and will be illustrated on an equation of a Kochen-Specker system. The orthogonality condition between vectors $j_1, j_2$ may be written as

$$\sum_{p=1}^{p=n} aj_1p\ aj_2p = 0 \tag{1}$$

This equation may be written as

$$\sum_{p=2}^{p=n} aj_1p\ aj_2p = -aj_11aj_21$$

Let $W_l, W_r$ be the interval evaluation of the left and right term of this equation. If the intersection $W$ of $W_l, W_r$ is empty then equation (1) cannot have a solution. Now assume that $W \subset W_r$ and that the interval for $aj_21$ does not include 0: then the interval for $aj_11$ must be included in $W/aj_21$: this may allow to reduce the range for $aj_11$. Clearly, the same method may be used for the other variables of the equation.

The same method may be used for the equations that describe that the vector are unitary. We have

$$\sum_{p=1}^{p=n} aj_1p^2 = 1$$

which may be written as

$$\sum_{p=2}^{p=n} aj_1p^2 - 1 = aj_11^2$$

Here again, if the intersection of $W_l, W_r$ has no intersection then there is no solution for this equation. Assume now that $W = [\underline{W}, \overline{W}]$ is a subset of $W_l$. The range for $aj_11$ must be included in

- $[-\sqrt{\overline{W}}, -\sqrt{\underline{W}}] \cup [\sqrt{\underline{W}}, \sqrt{\overline{W}}]$ if $\underline{W} > 0$
- $[-\sqrt{\overline{W}}, \sqrt{\overline{W}}]$ if $\underline{W} \leq 0$

The other filtering method that will be used is called the *3B method*. Let $\mathcal{I}_k$ be box that is considered for filtering, $[\underline{x_r}, \overline{x_r}]$ the range for the variable $x_r$ in the box and $\alpha$ a small number that is smaller than the width of $x_r$. Consider now a box $\mathcal{B}$ that is a copy of $\mathcal{I}_k$ except for the range $x_r$, which is substituted by $[\underline{x_r}, \underline{x_r} + \alpha]$. We then compute $F(\mathcal{B})$: if one of the interval vector in $F$ does not include 0 then we may modify the range for $x_r$ in $\mathcal{I}_k$ to $[\underline{x_r} + \alpha, \overline{x_r}]$ as the removed part of $\mathcal{I}_k$ cannot include a solution. We may then repeat the procedure but each time we double the value of $\alpha$ until no further reduction of $\mathcal{I}_k$ is obtained. The procedure may also be used to improve the upper bound of the range for $x_r$ (we test if the box with $[\overline{x_r} - \alpha, \overline{x_r}]$ as range for $x_r$ may include a solution) and evidently with the other variables. Note that the 2B method may be used to determine that there is no solution in the box $\mathcal{B}$.

Both the 2B and 3B method may be repeated as soon as there is a change in the width of one variable that may induce a change on another variable that has not been modified in a previous step. However, as the decrease of the widths of the ranges will usually become smaller and smaller, it will be more efficient at some point to use the bisection procedure.

The drawback of the 2B and 3B methods is that they are *local*: each equation may be consistent (i.e. no improvement will be obtained with these methods) but the whole system may be inconsistent. We use another filtering method that is *global*, namely the interval Newton method. Let $J$ be the square Jacobian matrix of the system of $m$ first equations chosen from the Kochen-Specker system. We define the interval vector $b$ as:

$$b = J^{-1}(Mid(\mathcal{I}_i))F(Mid(\mathcal{I}_i))$$

Note that although is may seem that $b$ is not an interval vector, the quantity $F(Mid(\mathcal{I}_i))$ is not a scalar vector as round-off errors have to be taken into account. On the other hand numerical errors in the inversion of the Jacobian matrix do not play a role. Now let us define:

$$V = JJ^{-1}(Mid(\mathcal{I}_i))$$
$$U = V(Mid(\mathcal{I}_i) - \mathcal{I}_i)$$

and the iterative scheme for the variable $x_k$ whose range is $\mathbf{x_k}$ as:

$$Z_k = \mathbf{x_k} + (U - b(k))/V(k,k)$$

assuming that the interval $V(k,k)$ does not include 0. The following result may be shown:

- if $Z_k \cap \mathbf{x_k} = \emptyset$, then there is no solution in $\mathcal{I}_i$
- if there is a solution in $\mathcal{I}_i$, then the value for $x_k$ will lie in $Z_k \cap \mathbf{x_k}$

### 4.2.2 Choosing the bisected variables

When using the bisection procedure, we have to choose which variable will have its range bisected. Classical heuristics are:

- bisecting the variable whose range has the largest width
- bisecting the variables in turn (called also the *round-robin* method)

The drawback of these methods is that they do not use the influence that the variables may have on the equations. Our bisection method is based on the *smear function* as defined by Kearfott [11]: let $J = ((J_{ij}))$ be the Jacobian matrix of the system and let define for the variable $x_i$ the *smear value* $s_i$ as:

$$\mathrm{Max}(|\underline{J_{ij}[\underline{x_i}, \overline{x_i}]}|, |\overline{J_{ij}[\underline{x_i}, \overline{x_i}]}|)$$

for all $j$ in $[1, l]$. The variable that will be bisected will be the one having the largest $s_i$.

### 4.2.3 Dealing with the non-collinearity constraints

As seen in the introduction, the non-collinearity constraint is not strict in practice. We may assume that the angle between two vectors cannot be smaller than a fixed threshold $\alpha$ (from a technological viewpoint this correspond, to a minimal angular distance between the measurement axis of two distinct detectors). Under that assumption the absolute value of the dot product between two vectors cannot be greater than $\cos\alpha$.

This constraint will be used in the filter. Interval arithmetic will be used to compute the interval evaluation of the dot product for a given box. If the lower bound of this evaluation is greater than $\cos\alpha$ or if the upper bound is lower than $-\cos\alpha$ the box will be eliminated.

### 4.2.4 Choosing the basis group

As seen previously, one of the groups in the Kochen-Specker system will be chosen as the basis of $R^n$. Let $\mathbf{X_i}$ be one of the vectors in this group, which will have only 0 components except for the $i$-th that will be 1. Assume now that this vector appears in another group including the vector $\mathbf{Y}$. The orthogonality condition between $\mathbf{X_i}$ and $\mathbf{Y}$ will imply that the $i$-th component of $\mathbf{Y}$ will be 0. Clearly, canceling vector components will have a positive impact on the solving of the system as it reduces the number of unknowns and simplifies the remaining equations. A natural heuristic to choose the basis group is to determine among all the groups which one will cancel the maximal number of vector components.

Consider for example Cabello's system:

```
1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH
```

Here, whatever is the chosen basis group, 12 vector components will be canceled. Hence all the groups are equivalent, and `1234` will be selected as basis group.

### 4.2.5 Finding the search space

As mentioned previously, any vector member of the solution set of a Kochen-Specker system may be substituted by its opposite and the solution set will still be a solution. This may be

used to decrease the size of the search space. Indeed, the range for the first component of all the vectors may be reduced to [0,1] instead of [-1,1], thereby eliminating the remaining multiplicative solution (except if a solution is obtained for vectors whose first component is exactly 0).

## 4.3 Decreasing the number of unknowns

Although the solver is able to deal with systems having a relatively large number of equations and unknowns, it may be interesting to decrease the number of unknowns. Different strategies for that purpose are presented in this section. Note, however, that these strategies may lead to equations that have multiple occurrences of the same variable (at the opposite of the initial set of equations): this may induce an overestimation of the interval evaluation of the equation and decrease the overall efficiency of the solver. Hence the right compromise between a reduced number of unknowns and more complex equations has to be found.

### 4.3.1 Combining equations in $R^4$

As seen previously, the filtering methods make use of each equation to reduce the search space for the unknowns. Hence additional equations derived from the initial system may be useful as long as they do not involve new variables. We will see that it will also be possible to use this additional equation to eliminate vectors. However, solving with a reduced set of variables will not allow us to verify that the eliminated vectors will satisfy the non-collinearity constraints. Hence it will be necessary afterward to calculate them and verify these constraints.

We have also seen that the choice of a basis group will allow us to determine that some components of the vectors are 0. Note first that a given vector cannot appear in groups that involve 3 or 4 vectors of the basis group: indeed, in that case the vector will be collinear with one of the vectors of the basis group or will be the zero vector. Hence we may assume that the vectors may have at most one or two zero components.

First let us assume that a given vector $\mathbf{X}$ has 2 zero components and let us denote by $x_1, x_2$ its nonzero components. As soon as this vector appears in a group the orthogonality condition will lead to 3 equations that involves $x_1, x_2$.

Now consider the set $\mathcal{S}_X$ of all equations in the Kochen-Specker system that involves only $x_1, x_2$ and every pair of equations in this set:

$$x_1 a_1 + x_2 a_2 = 0$$
$$x_1 b_1 + x_2 b_2 = 0$$

As $x_1, x_2$ cannot be simultaneously 0, we derive a necessary condition for this system to be satisfied:

$$a_1 b_2 - a_2 b_1 = 0 \tag{2}$$

Deriving this condition for all pairs of equations in $\mathcal{S}_X$ will allow us to obtain additional equations. Furthermore, as all these equations will involve different pairs of vectors, we may remove $x_1, x_2$ from the list of unknowns.

Let us furthermore that one of the vector $\mathbf{A}, \mathbf{B}$, whose components appears in the equation (2), has also two zero components. Let us assume that $a_3 = a_4 = 0$. The orthogonality condition between $\mathbf{A}, \mathbf{B}$ will be written as:

$$a_1 b_1 + a_2 b_2 = 0$$

This equation combined with equation (2) is a linear system in $a_1, a_2$. As $a_1, a_2$ cannot be simultaneously 0 we derive that

$$b_2^2 + b_1^2 = 0$$

from which we will deduce $b_1 = b_2 = 0$. Hence deriving the necessary condition will not only allow us to reduce the number of variables but also to determine the values of some vector components.

Now assume that the vector $\mathbf{X}$ has one zero component. The orthogonality equations will be obtained as:

$$\begin{aligned} x_1 a_1 + x_2 a_2 + x_3 a_3 &= 0 \\ x_1 b_1 + x_2 b_2 + x_3 b_3 &= 0 \\ x_1 c_1 + x_2 c_2 + x_3 c_3 &= 0 \end{aligned}$$

As $x_1, x_2, x_3$ cannot be simultaneously 0 we derive that:

$$\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = 0$$

Expanding this determinant will lead to a new equation. If all triplets of equations that involve $x_1, x_2, x_3$ are processed in this manner, then these variables may be removed. Instead of removing these variables we may proceed along a different path (that will work in any dimension). Assume that three vectors $\mathbf{W_i}, \mathbf{W_j}, \mathbf{W_k}$ in the same group (hence mutually orthogonal) lie in the same 3D space (i.e. they have $n-3$ identical zero components). Each of these vectors may be obtained as $\pm$ the cross-product of the two other vectors. The square of each component of this vector is therefore equal to the square of the component of the cross-product. These additional equations may be used with the 2B method to speed-up the solving procedure.

Note that there is a special case of this procedure that will not only lead to the elimination of variables but also to the determination of the value of one component of another vector. This will be illustrated by an example. Let us assume that the vector 9 has only two nonzero components, say $a93$ and $a94$, that the vectors 6 and 7 have one zero component,

say $a61, a71$, and that a group in the Kochen-Specker system is `6789`. The orthogonality condition between `679` is written as:

$$a63a93 + a64a94 = 0$$
$$a73a93 + a74a94 = 0$$

Using the method described previously we get the condition:

$$a63a74 - a64a73 = 0$$

Using the orthogonality condition between `6` and `8` we get:

$$a62a72 + a63a73 + a64a74 = 0$$
$$a62a82 + a63a83 + a64a84 = 0$$

We have now a system of three linear equations in the unknowns $a62, a63, a64$ which cannot be all 0. The determinant of this system leads to:

$$-a72(a73a83 + a74a84) + a82(a73^2 + a74^2) \tag{3}$$

Using the unitary condition on `7` we get

$$a73^2 + a74^2 = 1 - a72^2$$

Using this equation in (3) we get:

$$-a72(a73a83 + a74a84) + a82(1 - a72^2) \tag{4}$$

Now we use the the orthogonality condition on `7,8`

$$a72a82 + a73a83 + a74a84 = 0$$

Equation (4) become:

$$-a72^2a82 + a82(1 - a72^2) = a82 = 0$$

Hence we may assign the value 0 to $a82$.

### 4.3.2 Combining equations in $R^3$

For vectors in $R^3$, we may have only one zero component in a vector, otherwise the vector will be collinear with one vector of the basis group. The strategy used for the 4D vector may still be applied.

But additional equations may also be produced easily. Indeed, if a group is $\mathbf{X_i X_j X_k}$ then we may write $\mathbf{X_k} = \pm\mathbf{X_i} \times \mathbf{X_j}$. To deal with the sign change we will write the following equation for each component of $\mathbf{X_k}$:

$$xkm^2 = (xip\ xjq - xiq\ xjp)^2$$

Note that if $xkm$ is 0 then this equation simplifies to $xip\ xjq - xiq\ xjp = 0$.

## 4.4 Proving the existence of a solution

The solving algorithm as implemented does not allow us to prove exactly that a system has a solution but will only return interval solutions that are guaranteed to include a solution if there is any. We have now to investigate how to prove effectively that a system has a solution.

### 4.4.1 Manual proof

The first approach is based on our experience with solving of Kochen-Specker systems: most of the vector components of their solutions have values that are restricted by belonging to a given set $\mathcal{V}$:

- for dimension 4: $\mathcal{V} = \{ 0, \pm 1/3, \pm 2/3, \pm\sqrt{3}/3, \pm 1/2, \pm\sqrt{2}/2 \}$
- for dimension 3: $\mathcal{V} = \{ 0, \pm 1/3, \pm 2/3, \pm\sqrt{3}/3, \pm 1/2, \pm\sqrt{2}/2, \pm\sqrt{3}\sqrt{2}/6, \pm\sqrt{15}/5, \pm\sqrt{30}/6, \pm\sqrt{15}/15 \}$

At this time we have no theoretical explanation of this fact (except if we assume that the system has a solution in which all the vector components have a value that is either -1, 0 or 1, see the next section). But we may still use this property to prove that a system has effectively a solution.

We have designed a Maple program that takes as inputs the description of a Kochen-Specker system together with the interval solutions that have been found by the solver. Remember that the solver may have discarded some vectors during the pre-processing phase of section 4.3. The program first computes all the orthogonality and unitary equations without discarding any vector. For each unknown, if the mid-point of the corresponding interval solution is close to a value in the above set, we assign this value to this unknown. As soon as the unknown list has been processed we look at the equations that are not solved. Some of them may be linear in the unknowns: we solve this equation in one of the unknowns. This process is repeated until no additional unknown is determined.

At this stage we may have determined all the vectors: a Maple procedure then computes the absolute value of all the dot products between all pairs of vectors to verify the non-collinearity constraints. If none of these values is 1 then we have proved that the system has a solution.

It may also happen that some vector components are still undetermined. At this point we will look at the unitary equations. It may happen that some of these equations have only one unknown. Hence in that case the unknown may have two different values. We choose one of these values and assign it to the unknown. The orthogonality equations are then used to determine the values of additional unknowns. We repeat the process until all the unknowns have been determined. The non-collinearity test is then used. If it fails we backtrack: if an unknown value has been determined through a unitary equation, choosing one value among the two possible ones, then the opposite value will be assigned to the unknown.

Consider, for example, the Cabello's system:

$$\texttt{1234}, \texttt{4567}, \texttt{789A}, \texttt{ABCD}, \texttt{DEFG}, \texttt{GHI1}, \texttt{35CE}, \texttt{29BI}, \texttt{68FH}$$

The selected basis group is 1234, and the orthogonality condition may be written as:

$$a53\ a63 + a54\ a64 = 0 \qquad a53\ a73 + a54\ a74 = 0$$
$$a62\ a72 + a63\ a73 + a64\ a74 = 0 \qquad a72\ a82 + a73\ a83 + a74\ a84 = 0$$
$$a72\ a92 + a74\ a94 = 0 \qquad a72\ a102 + a73\ a103 + a74\ a104 = 0$$
$$a81\ a91 + a82\ a92 + a84\ a94 = 0 \qquad a81\ a101 + a82\ a102 + a83\ a103 + a84\ a104 = 0$$
$$a91\ a101 + a92\ a102 + a94\ a104 = 0 \qquad a101\ a111 + a102\ a112 + a104\ a114 = 0$$
$$a101\ a121 + a103\ a123 + a104\ a124 = 0 \qquad a101\ a131 + a102\ a132 + a103\ a133 + a104\ a134 = 0$$
$$a111\ a121 + a114\ a124 = 0 \qquad a111\ a131 + a112\ a132 + a114\ a134 = 0$$
$$a121\ a131 + a123\ a133 + a124\ a134 = 0 \qquad a131\ a141 + a133\ a143 + a134\ a144 = 0$$
$$a131\ a151 + a132\ a152 + a133\ a153 + a134\ a154 = 0 \qquad a131\ a161 + a132\ a162 + a133\ a163 = 0$$
$$a141\ a151 + a143\ a153 + a144\ a154 = 0 \qquad a141\ a161 + a143\ a163 = 0$$
$$a151\ a161 + a152\ a162 + a153\ a163 = 0 \qquad a161\ a171 + a162\ a172 + a163\ a173 = 0$$
$$a161\ a181 + a162\ a182 = 0 \qquad a171\ a181 + a172\ a182 = 0$$
$$a53\ a123 + a54\ a124 = 0 \qquad a53\ a143 + a54\ a144 = 0$$
$$a121\ a141 + a123\ a143 + a124\ a144 = 0 \qquad a91\ a111 + a92\ a112 + a94\ a114 = 0$$
$$a91\ a181 + a92\ a182 = 0 \qquad a111\ a181 + a112\ a182 = 0$$
$$a62\ a82 + a63\ a83 + a64\ a84 = 0 \qquad a62\ a152 + a63\ a153 + a64\ a154 = 0$$
$$a62\ a172 + a63\ a173 = 0 \qquad a81\ a151 + a82\ a152 + a83\ a153 + a84\ a154 = 0$$
$$a81\ a171 + a82\ a172 + a83\ a173 = 0 \qquad a151\ a171 + a152\ a172 + a153\ a173 = 0$$

The 32 variables used in the solver are:

```
[a72, a73, a74, a81, a83, a84, a101, a102, a103, a111, a112, a114,
a121, a123, a124, a131, a132, a133, a134, a141, a143, a144,
a151, a152, a153, a154, a161, a162, a163, a171, a172, a173]
```

and the solver provides the following interval solutions:

$$a72 = [.7070320229151, .70713178113013] \qquad a73 = [-.50004598260911, -.49997258438718]$$
$$a74 = [.49996812017253, .50004379793779] \qquad a81 = [.70707175486468, .70712250000001]$$
$$a83 = [.49994470302941, .50004808676851] \qquad a84 = [.49997940464194, .50004042079311]$$
$$a101 = [.49997227162018, .50005000000001] \qquad a102 = [-.50007650140248, -.49995045605404]$$
$$a103 = [-.70714999999999, -.70706590271862] \qquad a111 = [.49991, .50002000000001]$$
$$a112 = [.50007, .50013315611188] \qquad a114 = [.70704203035544, .70704877845822]$$
$$a121 = [.70711000000001, .70712000000001] \qquad a123 = [.50004671363637, .50006065240012]$$
$$a124 = [-.49998748682551, -.49992323851443] \qquad a131 = [0, .17031547791655e-6]$$
$$a132 = [-.7071152518822, -.707115] \qquad a133 = [.50001017745745, .50001554759612]$$
$$a134 = [.50007076877523, .50007481113807] \qquad a141 = [.7070717510339, .70707500000001]$$
$$a143 = [-.49997026594097, -.49996] \qquad a144 = [.49999258558033, .50003255023753]$$
$$a151 = [.49999994694038, .50000005315071] \qquad a152 = [-.50000064746846, -.49999935547389]$$
$$a153 = [.96339588551904e-5, .23249689161164e-4] \qquad a154 = [-.70710705468682, -.707106506802]$$
$$a161 = [.50000325872397, .50001213464297] \qquad a162 = [.50001563824114, .50001895126214]$$
$$a163 = [.70710667021816, .70710689243824] \qquad a171 = [.50001132991274, .50002113904678]$$
$$a172 = [.50000018905546, .50000735257214] \qquad a173 = [-.70710747465462, -.70710609126643]$$

Among these 32 interval, 31 are recognized as elements of $\mathcal{V}$. After having assigned these variables, the remaining equations are:

$$\begin{array}{ll}
a53\ a63 + a54\ a64 = 0 & -1/2\ a53 + 1/2\ a54 = 0 \\
1/2\ a62\ \sqrt{2} - 1/2\ a63 + 1/2\ a64 = 0 & 1/2\ \sqrt{2}\ a82 = 0 \\
1/2\ \sqrt{2}\ a92 + 1/2\ a94 = 0 & 1/2\ a104 = 0 \\
1/2\ \sqrt{2}\ a91 + a82\ a92 + 1/2\ a94 = 0 & -1/2\ a82 + 1/2\ a104 = 0 \\
1/2\ a91 - 1/2\ a92 + a94\ a104 = 0 & 1/4 - 1/2\ a112 + 1/2\ a104\ \sqrt{2} = 0 \\
-1/2\ a112\ \sqrt{2} + 1/4\ \sqrt{2} = 0 & 1/2\ a181 + 1/2\ a182 = 0 \\
1/2\ a181 + 1/2\ a182 = 0 & 1/2\ a53 - 1/2\ a54 = 0 \\
-1/2\ a53 + 1/2\ a54 = 0 & 1/2\ a91 + a92\ a112 + 1/2\ a94\ \sqrt{2} = 0 \\
a91\ a181 + a92\ a18 = 0 & 1/2\ a181 + a112\ a18 = 0 \\
a62\ a82 + 1/2\ a63 + 1/2\ a6 = 0 & -1/2\ a62 - 1/2\ a64\ \sqrt{2} = 0 \\
1/2\ a62 - 1/2\ a63\ \sqrt{2} = 0 & -1/2\ a82 = 0
\end{array}$$

Solving the linear equations in this set allow, to get

$$\begin{array}{lll}
a53 = a54 & a82 = 0 & a94 = -a92\sqrt{2} \\
a104 = 0 & a91 = a92 & a112 = 1/2 \\
a81 = -a182 & a63 = -a64 & a62 = -\sqrt{2}a64
\end{array}$$

The remaining unknowns are $a54, a64, a92, a182$. As these variables involve each only one vector, the unitary condition allows us to determine two possible values for each unknown. Assigning:

$$a54 = \sqrt{2}/2 \quad a64 = 1/2 \quad a92 = -1/2 \quad a182 = \sqrt{2}/2$$

allows to get a vector set that satisfies the non-collinearity condition.

# 5 Solving Kochen-Specker systems with Ritt's triangularization

Another approach for solving KS-system is based on symbolic processing. We will use a variant of Ritt's triangularization algorithm. To use this algorithm, we will assume that at least one of the vectors has $n-2$ zero components. The non-collinearity constraint implies that none of the remaining components may be 0. These components will appear in some of the orthogonality equations. For example when dealing with Cabello's system:

`1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH`

after having assigned `1234` as the basis group we get that `5`, `I` have two zero components. Vector `5` appears in the orthogonality equations:

$$a53\ a63 + a54\ a64 = 0$$
$$a53\ a73 + a54\ a74 = 0$$
$$a53\ a123 + a54\ a124 = 0$$
$$a53\ a143 + a54\ a144 = 0$$

As $a53$ cannot be 0, we may calculate $a63, a73, a123, a143$ as

$$a63 = -\frac{a54\ a64}{a53} \qquad a73 = -\frac{a54\ a74}{a53}$$
$$a123 = -\frac{a54\ a124}{a53} \qquad a143 = -\frac{a54\ a144}{a53}$$

Note that `6` is $[0, a62, a63, a64]$. From the above equations, we deduce that neither $a63$ nor $a64$ may be 0. Using the orthogonality equations involving `I` we may in the same way calculate $a111, a161, a171, a191$ and determine that some vector components cannot be 0.

A table $\mathcal{T}$ in the algorithm allows us to keep track of all the vectors components that cannot be 0.

The value of the vector components that have been determined at this step will be used in the remaining equations. A *cleaning* process will be applied on these equations:

- only the numerator of the equations will be considered
- the equations are factored
- if an equations is the product of several terms, factors that cannot be 0 will be eliminated. Such terms may be
  - vector components for which $\mathcal{T}$ indicates that the components cannot be 0
  - terms involving only the sum of square elements that cannot be 0. For example in the case of the Cabello's system an equation will be written as $a82\ a62\ (a74^2 + a72^2\ a53^2) = 0$ while it has been determined that neither $a53$ nor $a72, a74$ can be 0. Hence this equation will be substituted by the equation $a62\ a82 = 0$

We then proceed with an *update* of the table $\mathcal{T}$. We will assume in turn that each of the remaining vector components $aij$ are 0. As other components of vectors may be a function of $aij$ we will consider each vector and determine if setting $aij$ to 0 will not imply that a vector is collinear with another vector or a vector in the basis group (i.e. that setting $aij$ to 0 induces that a vector has at least 3 zero components) in which case $\mathcal{T}$ will be updated to indicate that $aij$ cannot be 0.

Then we proceed to a *determination* step. If the coefficient of a vector component in an orthogonality equation cannot be 0 we assign to this component the value that is derived from this equation. For example for Cabello's system the orthogonality equation between 7, 8 will be

$$a72\ a82\ a53 - a54\ a74\ a83 + a74\ a84\ a53 = 0$$

As $a54, a74$ cannot be 0 we derive

$$a83 = \frac{a53\ (a72\ a82 + a74\ a84)}{a54\ a74}$$

After having assigned this value we proceed with a cleaning step and an update step. This procedure is repeated while unknowns may be eliminated. Note that the cleaning process may lead to a univariate polynomial in one of the unknowns that may allow us to determine the value of the unknown. At this stage we use these polynomials only if they are of degree 1. During the process we may also encounter an inconsistency: typically we get an equation that requires one (or more) variables to be 0 while the zero-table indicates that this variable cannot be 0, or we get an equation that cannot be verified by real numbers.

When no more unknowns can be determined, we use another determination step that is basically similar to the previous one except that we deal now with the square of unknowns.

This process may determine that a KS system has no solution or on the other hand to allow us to determine all its solutions, although there is no guarantee of convergence for the process.

For example, for Cabello's system we get as our final result:

$$\begin{array}{ll} 2\ a54^2 - 1 = 0 & 2\ a53^2 - 1 = 0 \\ 2\ a62^2 - 1 = 0 & 2\ a94^2 - 1 = 0 \\ 4\ aA2^2 - 1 & -1 + 2\ aC1^2 = 0 \\ 4\ aE4^2 - 1 & -1 + 2\ aG3^2 \\ -1 + 2\ aI2^2 & -1 + 2\ aI1^2 = 0 \end{array}$$

Solving the above equations, we may determine all solutions of the system using the following vectors:

$$
\begin{aligned}
&\texttt{5} : [0, 0, a53, a54] \\
&\texttt{6} : [0, a62, -2\ a54\ a62\ a94\ a53, a62\ a94] \\
&\texttt{7} : [0, -a94, -a54/(2\ a53), 1/2] \\
&\texttt{8} : [aI1/(2\ a94\ aI2), 0, a53/(2\ a54), 1/2] \\
&\texttt{9} : [-aI2/(2\ aI1), 1/2, 0, a94] \\
&\texttt{A} : [aI1\ aA2/aI2, aA2, -2\ a53\ a94\ aA2/a54, 0] \\
&\texttt{B} : [-aI2/(2\ aI1), 1/2, 0, -1/(2\ a94)] \\
&\texttt{C} : [aC1, 0, 2\ a54\ a94\ aI2\ aC1\ aI1/a53, -2\ a94\ aI2\ aC1\ aI1] \\
&\texttt{D} : [0, 1/(2\ a94), 1/(4\ a54\ a53), 1/2] \\
&\texttt{E} : [aE4\ a94/(aI2\ aI1), 0, -a54\ aE4/a53, aE4] \\
&\texttt{F} : [aI2\ aI1, 1/2, 0, -1/(2\ a94)] \\
&\texttt{G} : [aG3\ aI1\ aI2/(2\ a54\ a94\ a53), -aG3/(4\ a54\ a53\ a94), aG3, 0] \\
&\texttt{H} : [a54\ a94\ a53\ \sqrt{2}\ aI2/aI1, -a54\ a94\ a53\ \sqrt{2}, -\sqrt{2}/2, 0] \\
&\texttt{I} : [aI1, aI2, 0, 0]
\end{aligned}
$$

# 6 Systems with $\{-1, 0, 1\}$ solutions

We will investigate systems that admit at least one solution for which all the components of all vectors are in the set $\{-1, 0, 1\}$ (such a solution will be denoted a $\{-1, 0, 1\}$ solution).

Let us now consider with an n=4 example whose normalized solutions are unitary. The solutions, i.e., unitary vectors can be classified into one of these categories:

- *category 1*: a vector with 3 zero components. As the vector is unitary the remaining component's value will be $\pm 1$
- *category 2*: a vector with 2 zero components. The two nonzero components must then be identical up to their sign. Let $a$ denote the absolute value of the nonzero components. The norm of such vector will be $2a^2 = 1$ and hence $a = \sqrt{2}/2$ and the nonzero components will be $\pm a$
- *category 3*: a vector with 1 zero component. Using the same reasoning than above the absolute value $a$ of the nonzero components will be $a = \sqrt{3}/3$
- *category 4*: a vector with no zero component. The absolute value of the component will be $a = 1/2$ and such a vector will be written $[\pm a, \pm a, \pm a, \pm a]$

Hence there is a finite set $\mathcal{CN}$ of normalized vector that may be a normalized $\{-1,0,1\}$ solution of a system. This set has exactly 80 vectors.

Let us consider now all the solutions $\mathcal{S} = \{S_1, \ldots, S_n\}$ of the continuous system obtained by assuming that one group represents the orthogonal unitary basis of $R^4$. A system will have $\{-1,0,1\}$ solutions if there is a rotation matrix $\mathcal{R}$ that transforms at least one solution $S_i$ into a normalized $\{-1,0,1\}$ solution.

**Theorem A-6**: There is only a finite set of rotation matrices that may transform a continuous solution into a $\{-1,0,1\}$ solution,

**Proof**: Each column of the rotation matrix represents the normalized $\{-1,0,1\}$ solution for one of the unit vectors of the basis of $R^4$. Since the number of such normalized $\{-1,0,1\}$ solutions is finite, there is only a finite number of possible rotation matrices □

To determine these matrices we calculate all the matrices that may be obtained by considering all possible normalized $\{-1,0,1\}$ solutions as columns of the rotation matrix. We then check whether all the column are orthogonal and if the determinant of the matrix is 1. If these conditions are fulfilled, then the matrix is a rotation matrix. This leads to 6144 possible rotation matrices, but it must be noted that for each possible rotation matrix its opposite is also included in the set. Hence the set may be decomposed into two subsets $\mathcal{R}_1, \mathcal{R}_2$ in which $\mathcal{R}_2$ includes the opposite of the matrix in $\mathcal{R}_1$.

To test if a continuous solution may be transformed into a normalized $\{-1,0,1\}$ solution, it is thus sufficient to determine if there is at least one matrix in $\mathcal{R}_1$ that transforms the continuous solution into a $\{-1,0,1\}$ solution.

Consider now a system $\mathcal{S}$ and a set $\mathcal{N}$ of normalized vectors, including a basis of $R^4$, that is solution of $\mathcal{S}$. If the systems $\mathcal{S}$ admits a set of solution vectors that have only components in the set $\{-1,0,1\}$ then we have:

**Theorem B-6**: A vector in $\mathcal{N}$ can belong only to a finite set $\mathcal{U}_n$ of normalized vectors.

**Proof:** Consider the set $\mathcal{CN}$ of normalized $\{-1,0,1\}$ solutions of $\mathcal{S}$ and a solution $C_i$ of $\mathcal{S}$ in this set. A continuous solution $\mathcal{S}_i$ of $\mathcal{S}$ including a basis of $R^4$ that is identical to $C_i$ will be obtained by multiplying $C_i$ by a rotation matrix $\mathcal{R}$ such that $\mathcal{R}C_i$ includes a basis of $R^4$. This is possible only if each row of $\mathcal{R}$ corresponds to a vector in a group in this $C_i$.

The possible continuous solutions are thus obtained by considering all matrices whose rows are elements of $\mathcal{CN}$, rejecting the ones that cannot be rotation matrices and then multiplying these rotation matrices by all the elements in $\mathcal{CN}$. As $\mathcal{CN}$ is a finite set, the obtained vector set will also be finite □

The set of possible rotation matrices $\mathcal{R}$ may be easily obtained as the set of transposed matrices obtained from the matrices in $\mathcal{R}_1, \mathcal{R}_2$. After multiplying this set by the 80 vectors of $\mathcal{CN}$ we get that there are exactly 960 vectors in $\mathcal{U}_n$ and that the components of these

vectors each have a value in a set $\mathcal{V}$ of 21 values, namely:

$$
\begin{array}{l}
0, 1, -1 \\
1/2\ \sqrt{2}, -1/2\ \sqrt{2} \\
1/3\ \sqrt{3}, -1/3\ \sqrt{3} \\
1/2, -1/2 \\
1/3\ \sqrt{2}\ \sqrt{3}, -1/3\ \sqrt{2}\ \sqrt{3} \\
1/6\ \sqrt{2}\ \sqrt{3}, -1/6\ \sqrt{2}\ \sqrt{3} \\
1/6\ \sqrt{3}, -1/6\ \sqrt{3} \\
1/2\ \sqrt{3}, -1/2\ \sqrt{3} \\
2/3, -2/3 \\
1/3, -1/3
\end{array}
$$

## 6.1 Determining if a system has a $\{-1, 0, 1\}$ solution

To each vector $W$ in $\mathcal{U}_n$ we may associate a set $S_W$ of vectors that are perpendicular to $W$.

When considering a system and after assigning one of the group to be a basis of $R^4$, we look at each group in which one of the basis vectors is involved. Each vector of such a group that is not a basis vector may belong only to the set $S_W$ associated with the vector basis. We may then make a hypothesis for the first vector by assigning it to be one element of $S_W$. This reduces the possible values for the other vector of the group. All these constraints are propagated for each group until either an inconsistency is detected (two vectors have coordinates that are either identical (or opposite) or one orthogonality constraint is violated) or the system is consistent. In the former case we backtrack an assign another value for one of the vectors (if all values have been tested, then the system has no $\{-1, 0, 1\}$ solution). In the later case if all vectors have been assigned, then we have found a $\{-1, 0, 1\}$ solution, otherwise we assign one value to one of the unknown vectors (a value that respect the orthogonality constraints for each group in which the vector is involved) and we restart.

This algorithm allows us to determine very efficiently if the system has a $\{-1, 0, 1\}$ solution: the computation time is less than one second.

## 6.2 Fast algorithm for preselected vector component values

If we assume definite values for vector components, it is possible to compute very efficiently a solution for a system represented by an MMP diagram when a solution with those values exists. To this end N. Megill wrote a program called that finds solutions for several predetermined value sets. This program is typically very fast (less than 1 second on a PC). In addition to its use in trying to find simple solutions quickly, most of the final results described in Section 8 were independently verified with this program.

The speed is achieved in part by pre-computing all combinations of scalar products of allowable vector assignments. This is possible because the set of values is a small finite number.

The main algorithm scans the vectors and tries to assign unique vectors to them so that all vectors assigned to a given group are orthogonal. In case of a conflict the algorithm backtracks, until either all possible assignments have been exhausted or a solution is found.

By default, the algorithm is hard-coded to look for solutions with vector component values from set $\{-5,-2,-1,0,1,2,5\}$, although the values $\{-\sqrt{2},\sqrt{2}\}$ may be added to this list.

In the worst case, the behavior of the backtracking algorithm is exponential. We are able to attenuate that behavior in most cases by scanning next those vectors most tightly coupled to those already scanned. This helps to force conflicts to show up early on so that backtracking can take care of them more quickly. The algorithm for determining the scanning order is similar to the cluster presorting algorithm described in Section 2.2.

Even with the presorted scanning order, occasionally we encounter exponential behavior that this method does not attenuate. Therefore the program has a user-settable timeout parameter. Those few MMP diagrams that do not complete can be rerun later, if desired, with a larger setting.

# 7 Implementation

## 7.1 The preliminary pass

Our experience is that most of the Kochen-Specker systems will have no solution and that the majority of such systems may be eliminated by symbolic manipulation of the equations of the conditioned systems.

Since symbolic manipulation was involved we developed first a prototype in Maple for $n = 4$ that implemented the following steps:

1. choose the basis group using the heuristic described in section 4.2.4
2. determine which vector $\mathbf{X}$ appears in groups involving vectors of the basis group. The orthogonality condition between such a vector and the basis group vector will allow us to set to 0 some components of $\mathbf{X}$
3. compute the set $\mathcal{O}$ of all the orthogonality equations
4. if in $\mathcal{O}$ we have equations that are of type $ajk\ alk = 0$ corresponding to the orthogonality of vectors $\mathbf{X}_j, \mathbf{X}_l$, count the number $n_l, n_j$ of 0 components in these vectors
   - if $n_l = 2$ and $n_j < 2$ set $ajk$ to 0. Indeed, $alk$ cannot be 0, otherwise $\mathbf{X_l}$ will be collinear with a vector in the basis group. Similarly, if $n_j = n-2$ and $n_l < n-2$ we set $alk$ to 0

- if $n_l = 2$ and $n_j = 2$ then the system cannot have a solution, because satisfying the orthogonality condition would imply the collinearity of the vector with a vector of the basis group

5. consider all vectors that already have two 0 components. For a given vector $\mathbf{X}_k$ consider all pairs of equations in $\mathcal{O}$ involving $\mathbf{X}_k$. Each of these pairs will describe an orthogonality condition between $\mathbf{X}_k$ and two vectors $\mathbf{X_l}, \mathbf{X_j}$, and the equations may be written as:

$$aki\ ali + akm\ alm = 0$$
$$aki\ aji + akm\ ajm = 0$$

As $aki, akm$ cannot simultaneously be 0 we deduce the constraint $ali\ ajm - aji\ alm = 0$. As the vectors $\mathbf{X}_k, \mathbf{X}_l, \mathbf{X}_j$ should belong to the same group, we consider now the orthogonality condition between $\mathbf{X}_l, \mathbf{X}_j$. Assume that this orthogonality condition is written as $ali\ aji + alm\ ajm = 0$. We have therefore

$$ali\ ajm - aji\ alm = 0$$
$$ali\ aji + alm\ ajm = 0$$

The particular shape of the orthogonality condition implies that one of the vectors $\mathbf{X}_l, \mathbf{X}_j$, say $\mathbf{X}_l$, already has a 0 component. Hence $ali, alm$ cannot simultaneously be 0, and we deduce the constraint $ajm^2 + aji^2 = 0$ which implies that $ajm = aji = 0$. Note that if both $\mathbf{X}_l, \mathbf{X}_j$ already have one 0 component, we may eliminate the system as these vectors will be collinear with the basis group

6. if during the process one component is set to 0, then we update the orthogonality equations and restart at step 4

Although the Maple prototype was efficient it was able to deal with only about 10 Kochen-Specker systems per minute. In view of the large number of systems that were expected to be processed, the computation time was too large. Hence we decided to develop a specific C program that implemented the above manipulations. In view of the efficiency gain of this program, we decided that instead of using only one basis group we may assign every group in a system to be a basis group (although the group provided by the heuristic is always tested first).

The speed-up factor is considerable: the C program is able to deal with about 2.5 million systems per minute (150 millions per hour) on a single laptop. Splitting the Kochen-Specker file and using a cluster of 15 machines has allowed us to deal with about 2.25 billion systems per hour.

In view of this efficiency, the weak point of the whole process for relatively small values for $a, b$ was no longer the verification of the solution but instead the generation of the Kochen-Specker systems. Indeed, although the exhaustive generation of the systems was done on

a large cluster, we experienced difficulties for large values of $a$. To speed-up the process, Brendan McKay modified its generation program so that the Kochen-Specker systems were generated incrementally: after the generation of the first $m$ initial groups of a system, only the systems starting with these initial groups are generated until one group in 1, $m$ must be modified. This allows one to implement a pruning mechanism: if the process determines that the system with $m$ groups has already no solution, then the systems starting with the $m$ initial groups are not generated and the next initial group is considered. Clearly, our C program was the candidate of choice for this pruning mechanism.

For example, for $a = 10, b = 12$ without filtering, we generated 197 885 058 Kochen-Specker systems in about 5 hours while for $a = 11, b = 12$ we were not able to generate all the systems. In both cases the modified program determined almost immediately that there was no potentially valid system. For $a = 18, b = 12$ we were able to get all possible 100220 systems in about 16 hours while without filtering, over 2.9 $10^16$ systems would have been generated (although this would have required over 30 millions years on 2 GHz CPU).

The pruning mechanism also allows us to use a distributed implementation for the generation of the Kochen-Specker systems. In a first pass we will determine what may be the $m < a$ first groups of the sequence (typically $m$ will be 6). As soon as a set of $m$ groups is generated, it is saved in a file, and the pruning mechanism is used to discard any systems that start with the same $m$ initial groups. Then a set of slave computers is used to generate all the sequences that begin with a set of $m$ groups as found in the file, discarding all other systems.

In dimension 3 such an approach is no longer valid since similar simplification rules cannot be derived.

## 7.2 The "hard" solver

After the generation and the filter through the preliminary pass, the systems that are of interest remain. Now we will use the solving algorithm described in section 4 to determine if some of these systems may have at least one real solution. As the number of systems may still be relatively large our objective is to design a fully automated software process.

Fortunately, we have the right tool to design such a procedure with the `ALIAS` library[2]. The kernel of this library is a collection of solving algorithms written in C++ that use the BIAS/Profil interval arithmetic. The algorithm described in section 4 is implemented as a generic solving procedure: a C++ procedure is designed to handle any system. The arguments of this procedure are basically the number of unknowns and the number of equations and two key procedures:

- an `F` procedure that will allow the interval evaluation of the equations for a given box
- a `J` procedure that computes the interval evaluation of the components of the Jacobian matrix of the system

[2] see `www.inria.fr/coprin/logiciel/ALIAS`

Furthermore, the solving procedure may have an optional argument which is a C++ program provided by the user that will allow, to filter the boxes i.e. to reduce the width of a box or even to determine that there is no solution in a given box. This possibility will be used to implement the 2B method described in section (4.2.1) and a filter for the non-collinearity constraints.

An advantage of the `ALIAS` library is that the solving algorithm described in section (4) is fully interfaced with Maple. We have therefore designed a Maple procedure that reads the description of Kochen-Specker systems and proceeds along the following steps:

1. determine of the best basis group according to the strategy presented in section (4.2.4)
2. calculate of the orthogonality and unitary equations
3. eventually decrease the number of unknowns using the strategy presented in section (4.3)
4. generate the C++ filtering code (see section (4.2.1) that corresponds to the 2B method (the 3B method is already available within the C++ solving algorithm) using a Maple procedure that is part of the `ALIAS`-Maple library
5. generate of the C++ code that will allow us to filter out boxes that have vector components that will not fulfill the non-collinearity constraints. For that purpose we compute the interval evaluation of the dot product for each pair of vectors and if either the lower bound of this evaluation is greater than 0.99 or the upper bound is lower than -0.99, then the box is discarded
6. call a specific `ALIAS`-Maple procedure that will generate all the C++ code necessary for the solving procedure. The `F, J` C++ procedure will be automatically generated together with a main program. It will then be compiled and run and the result of the solving algorithm will be available directly within Maple

A feature of the `ALIAS` Maple interface is that it allows a distributed implementation of the solving algorithm. Indeed, it may be seen in the algorithm that the processing of one box in the list does not depend upon the other boxes in the list. Hence we may use the master-slave paradigm: a master program manages the list of boxes and sends one box to a slave computer. This slave computer will perform a limited number of bisections on this box and returns to the master program the remaining boxes together with the eventual solution that will be found by the slave. Communication between the slave and the master uses the PVM message passing software. In our case, distributed implementation may allow us to reduce drastically the computation time. Indeed, when using a single computer, the solutions of the system may be included in the last box of the list of boxes. Using a distributed implementation this box may be processed very early in the algorithm. Hence the decrease in the computation time may be larger than expected from the number of slave computers. A distributed version may be generated directly within Maple by just providing a list of machine names together with how many boxes they may return.

The solving time is in general relatively small. For Cabello's system involving 32 unknowns and 82 equations, the computation time on a single laptop is about 60 seconds. For Kernaghan's system we have 35 unknowns and 81 equations, and the system is solved in about 75 seconds. Eliminating systems without solution is usually faster. For example, the system

`1234,1256,1378,129A,24BC,34DE,5DFG,7HIJ,8BFK,9ADE,CGHI,6EJK`

with 59 equations and 26 unknowns is eliminated in 10 seconds.

In dimension 3, Peres's system has 82 equations with 72 unknowns and is solved in 135 seconds, while Bub's system with 89 equations and 76 unknowns is solved in 78 seconds.

# 8 Results

For the systems in $R^4$, we have been able to show the following results for systems with a minimal loop of 3 blocks:

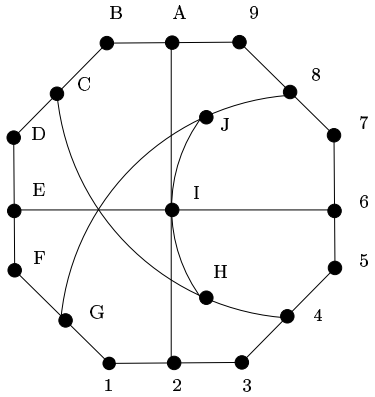
- Cabello's system with $a = 18, b = 9$ is the smallest 4-dim Kochen-Specker real system (figure 1)
- a general feature we found to hold for all MMP diagrams without `0-1` states we tested is that the number of edges, $b$ and the number of vertices that share more than one edge, $a^*$ satisfy the following inequality: $nb \geq 2a^*$, where $n$ is the number of vertices per edge. Hence, there are no KS vectors that share at least 3 of $b$ $n$-tuples in their KS set whose number $a^* > \frac{nb}{2}$. In $\mathbb{R}^n$ this means that we cannot arrive at systems with more unknowns than equations when we disregard the unknowns that appear in only two equations. To prove the feature for an arbitrary $n$ remains an open problem
- the smallest MMP diagrams with no `0-1` states are
  - *with 3 vertices per edge*

    7 vertices—5 edges (smallest loops of size 3): `123,345,561,275,476` (triangle);

    15-11 (4): `123,345,567,789,9AB,BC1,CD6,2DA,2E8,4FA,CEF` (hexagon),

    `123,345,567,789,9AB,BCD,DE1,4AE,28C,2FA,6FD` (heptagon);

    19-13 (5): `123,345,567,789,9AB,BCD,DEF,FG1,2IA,6IE,4HC,8JG,HIJ` (octagon) (2d);




`123,345,567,789,9AB,BCD,DE1,EI7,2F9,4GB,IJG,FJH,CH6` (heptagon);

– *with 4 vertices per edge* (see figure 2)

6-3 (smallest loops of size 2): `1234,2356,1456` (twisted triangle) (2a);

10-5 (smallest loops of size 3): `1234,4567,7891,35A8,29A6` (triangle) (2b);

22-11 (smallest loops of size 4):

`1234,4567,789A,ABCD,DEFG,GHI1,FJK5,HJMC,3KL8,IBL6,29ME.` (hexagon) (2c);

`1234,4567,789A,ABCD,DEF1,FGH5,EMJ6,2GLC,3IJ8,HIKB,MLK9` (pentagon);

38-19 (5):

`1234,1567,289A,5BCD,8BEF,3GHI,6JKL,GJMN,CHOP,EMQR,OQST,RUVW,`
`4UXY,9SZa,FIbc,KTXb,7VZc,ALPW,DNYa  (dodecagon)`

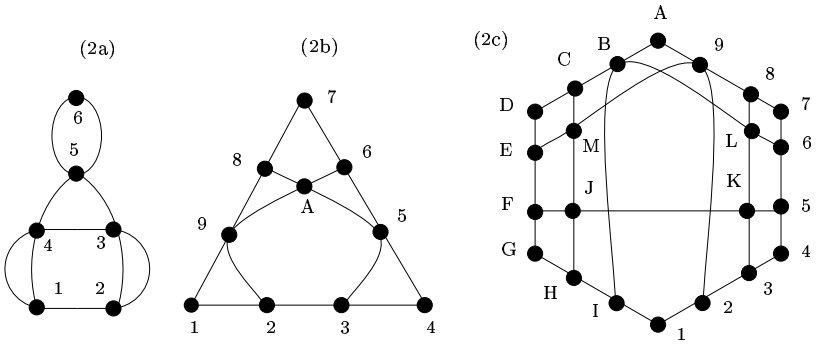
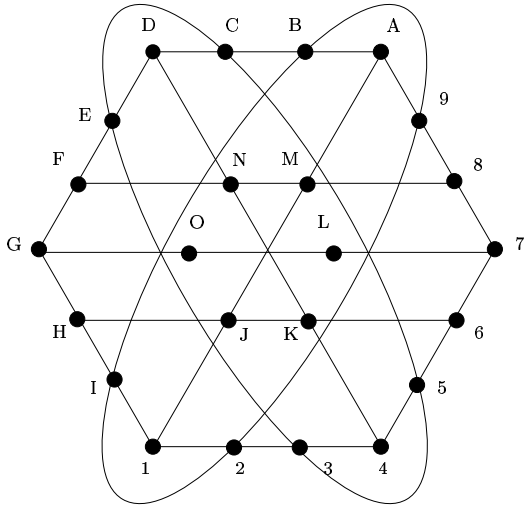


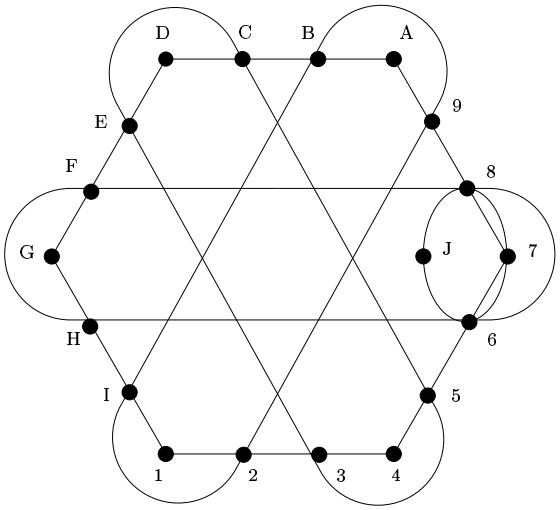

Figure 2: smallest diagrams with no `0-1` states and 4 vertices per edge

but none of them (except possibly the last one—see the end of this section) has a solution;

- the smallest KS vectors that we found to have real solutions are
  - Cabello `1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH` with $a = 18$, $b = 9$; it has solutions from the set $\{-1, 0, 1\}$;
  - `1234,4567,789A,ABCD,DEFG,GHI1,FNM8,GOL7,HJK6,DNK4,AMJ1,35CE,B29I` with $a = 24$, $b = 13$ which is the first system with smallest loops of size 3 that does not contain Cabello's; it does not have solutions from the set $\{-1, 0, 1\}$;



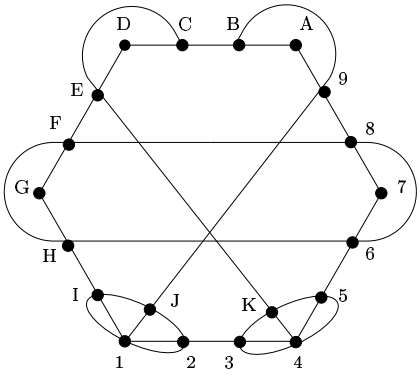

  - `1234,4567,789A,ABCD,DEFG,GHI1,35CE,29BI,68FH,678J` with $a = 19$, $b = 10$ and the smallest loop of size 2 that contains Cabello's and is the only diagram with 19 vectors that has a $\{-1, 0, 1\}$ solution

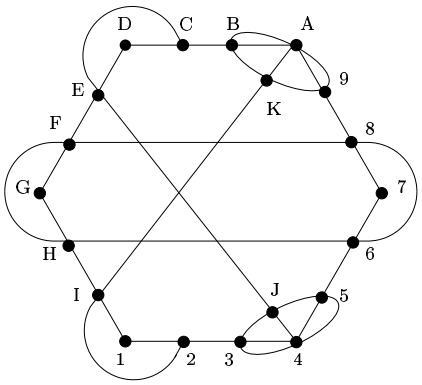


- the smallest 4-dim systems with loops of size 2 that do not contain Cabello's and do have $\{-1, 0, 1\}$ solutions are
  - `1234,4567,789A,ABCD,DEFG,GHI1,68FH,12JI,1J9B,345K,4KEC` (4a)
  - `1234,4567,789A,ABCD,DEFG,GHI1,68FH,2IAK,345J,4JEC,9ABK` (4b)

  with $a = 20$, $b = 10$, the last one being isomorphic to Kernaghan's system

(4a)

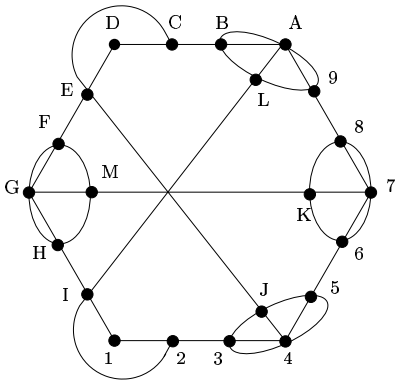


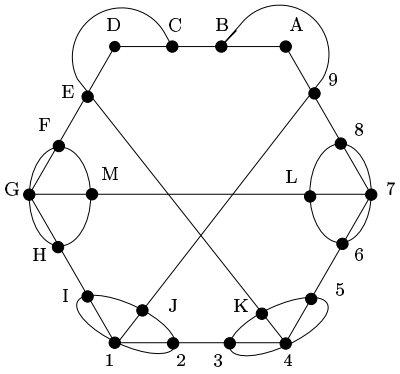




(4b)

- the smallest 4-dim systems with loops of size 2 that contain neither the two previous systems nor Cabello's system and do have $\{-1, 0, 1\}$ solutions are
  - `1234,4567,789A,ABCD,DEFG,GHI1,2ILA,345J,4JEC,678K,7KMG,9ABL,FGHM`, $a = 22$, $b = 13$; (4c)
  - `1234,4567,789A,ABCD,DEFG,GHI1,12IJ,345K,678L,GML7,1J9B,4KEC,FGHM`, $a = 22$, $b = 13$; (4d)

(4c)




(4d)

These two systems are the only such systems for $a = 22$ and they are not contained in any other system with $a = 22$

- all 4-dim systems with up to 22 vectors and 12 edges with the smallest loops of size 2 that do have solutions from $\{-1, 0, 1\}$ contain at least one of the systems (4a), (4b) and in many cases Cabello's system
- 4-dim systems with more than 41 vectors cannot have solutions with $\{-1, 0, 1\}$ components, and there are no such solutions to systems with minimal loops of size 5 up to 41 vectors
- 3-dim systems with loop of size 3 and 4 cannot have a real solution
- all 3-dim systems with up to 30 vectors and 20 groups have been scanned and no Kochen-Specker system was found
- in 3-dim systems, vectors that appear in only one group cannot be neglected and therefore Bub's system:

  `123,345,167,AB6,AC4,DEG,DFH,F9O,E8V,5JI,7MN,GIa,HNh,7LT,5KR,DAe,UTS,`
  `PRS,1GP,3HU,3Vj,Pgh,Uba,1Oi,VZg,OYb,6Xk,4Wn,Sde,dci,dfj,imn,jlk,akQ,`
  `hnQ,eQ2`

  is currently the smallest known 3-dim system. It contains 49 vectors

As an additional result, we may probably claim that we have established a world record in terms of number of equation systems that have been solved for a given problem: indeed, the number of systems that have been solved up to now is about 250 000 000. Generation of all systems with loops of size 2 has been performed by M. Pavičić on two clusters with 43 and 60 CPUs, while the solving part takes place either on a laptop or, for the most difficult tasks, on a cluster with 20 CPUs.

Still we have difficulty dealing with larger $a, b$ because:

- the generation time becomes very large
- the solving algorithm still fails for very large values of $a$. For example, we have failed to determine if the sequence:

  `1234,1567,289A,5BCD,8BEF,3GHI,CJKL,GJMN,4OPQ,6RST,ORUV,HWXY,7WZa,4Kbc,`
  `Zbde,DIUd,9LVX,ESYc,AMTe,FNOa`

  with $a = 40, b = 20$ has a solution.

We are now investigating the use of *grid computing* to deal with such large systems.

# 9 Conclusion

We have presented a constructive and exhaustive definition of Kochen-Specker systems. Our constructive definition of such systems is based on algorithms that generate linear MMP diagrams, on algorithms that filter out diagrams on which algebraic `0-1` states cannot be defined, and on algorithms that solve nonlinear equations describing the orthogonality of the vectors by means of polynomially complex interval analysis. To demonstrate the power of the algorithms, all 4-dim Kochen-Specker vector systems containing up to 24 vectors have been generated and described, all 3-dim vector systems containing up to 30 vectors have been scanned, and several general properties of Kochen-Specker systems have been found.

The presented algorithms can easily be generalized beyond Kochen-Specker systems: for example, one can use MMP diagrams to generate Hilbert lattice counterexamples and partial Boolean algebras.

# References

[1] Aravind P. K. Generalized Kochen-Specker theorem. *Phys. Rev. A*, **68**:051204–1–3, 2003. arXiv.org/abs/quant-ph/0301074.

[2] Bub J. Schütte's tautology and the Kochen-Specker theorem. *Found. Phys.*, 26:787–806, 1996.

[3] Cabello A. Kochen-Specker theorem and experimental tests on hidden variables. *Int. J. Mod. Phys. A*, **15**:2813–2820, 2000. arXiv.org/abs/quant-ph/9911022.

[4] Cabello A. Finite precision measurement does not nullify the Kochen-Specker theorem. *arXiv.org/abs/quant-ph/0104024*, 2001.

[5] Cabello A. Kochen-Specker theorem for a single qubit using positive operator-valued measures. *Phys. Rev. Lett.*, **90**:190401–1–4, 2003. arXiv.org/abs/quant-ph/0210082.

[6] Cabello A., Estebaranz J.M, and Garcia Alcaine G. Bell-Kochen-Specker theorem: A proof with 18 vectors. *Phys. Lett. A 212*, pages 183–187, 1996.

[7] Cabello A. and Garcia Alcaine G. Proposed experimental tests of the Bell-Kochen-Specker theorem. *Phys. Rev. Lett. 80*, pages 1797–1799, 1998.

[8] Collavizza M., F. Deloble, and Rueher M. Comparing partial consistencies. *Reliable Computing*, 5:1–16, 1999.

[9] Huang Y.F. and others . Kochen-Specker theorem for finite precision spin-one measurement. *Phys. Rev. Lett. 88*, pages 1–4, 2002.

[10] Hultgren, III Bror O. and Shimony A. The lattice of verifiable propositions of the spin–1 system. *J. Math. Phys.*, **18**:381–394, 1977.

[11] Kearfott R.B. and Manuel N. III. INTBIS, a portable interval Newton/Bisection package. *ACM Trans. on Mathematical Software*, 16(2):152–157, June 1990.

[12] Kent A. Noncontextual hidden variables and physical measurements. *Phys. Rev. Lett.*, **83**:3755–3757, 1999. arXiv.org/abs/quant-ph/9906006.

[13] Kernaghan M. Bell-Kochen-Specker Theorem for 20 vectors. *Journal of Physics A 27*, pages L829–L830, 1994.

[14] Kernaghan M. and Peres A. Kochen-Specker theorem for eightdimensional space. *Phys. Lett. A 198*, 1:1–5, 1995.

[15] Kochen S. and Specker E. P. The problem of hidden variables in quantum mechanics. *J. Math. Mech.*, **17**:59–87, 1967.

[16] McKay B.D., Megill N.D., and Pavičić M. Algorithms for Greechie diagrams. *Int. J. Theor. Phys.*, 39(10):2381–2406, 2000. arXiv.org/quant-ph/0009039.

[17] Mermin N.D. A Kochen-Specker theorem for imprecisely specified measurement. *arXiv.org/abs/quant-ph/9912081*, 1999.

[18] Meyer D.A. Finite precision measurement nullifies the Kochen-Specker theorem. *Phys. Rev. Lett.*, **83**:3751–3754, 1999. arXiv.org/abs/quant-ph/9905080.

[19] Pavičić M. Quantum computers, discrete space, and entanglement. In Callaos Nagib, He Yigaang, and Perez-Peraza Jorge A., editors, *SCI 2002/ISAS 2002 Proceedings, The 6th World Multiconference on Systemics, Cybernetics, and Informatics*, volume XVII, SCI in Physics, Astronomy and Chemistry, SCI, Orlando, Florida, 2002, pp. 65–70.

[20] Pavičić M. Constructing quantum reality. In Boniolo G. and Furlan G., editors, *The Role of Mathematics in Physical Sciences*, [to appear].

[21] Pavičić M. Kochen-Specker algorithms for qubits. In Stephen M. Barnett, Erika Andersson, John Jeffers, Patrik Öhberg, and Osamu Hirota, Editors, *Quantum Communication, Measurement and Computing: The Seventh International Conference on Quantum Communication, Measurement and Computing held in Glasgow, United Kingdom, 25-29 July 2004*, American Institute of Physics Conference Proceedings 734, 2004, pp. 195–198.

[22] Pavičić M. and Megill N.D. Equations, states, and lattices of infinite-dimensional Hilbert space. *Int. J. Theor. Phys.*, **39**:2349–2391, 2000. arXiv.org/abs/quant-ph/0009038.

[23] Pavičić M., Merlet J.-P., McKay B.D., and Megill N.D. Kochen-Specker vectors space. *J. Phys. A*, **37**: [submitted], 2004. arXiv.org/abs/quant-ph/0409014.

[24] Penrose R. On Bell non-locality without probabilities: Some curious geometry. In Ellis John and Amati Daniele, editors, *Quantum Reflections*, pages 1–27. Cambridge University Press, Cambridge, 2000.

[25] Peres A. Two simple proofs of the Bell-Kochen-Specker theorem. *J. Phys. A*, 24:L175–L178, 1991.

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

[27] Simon C., C. Brukner, and Zeilinger A. Hidden-variable theorems for real experiments. *Phys. Rev. Lett.*, **86**:4427–4430, 2001. arXiv.org/abs/quant-ph/0006043.

[28] Simon C. and others . A feasible Kochen-Specker experiment with single particles. *Phys. Rev. Lett. 85*, page 1783, 2000.

[29] Smith D. An orthomodular Bell-Kochen-Specker theorem. *Int. J. Theor. Phys.*, **43**:[to appear], 2004.

[30] Svozil K. and Tkadlec J. Greechie diagrams, nonexistence of measures and Kochen-Specker-type constructions. *J. Math. Phys.*, **37**:5380–5401, 1996.

[31] Tkadlec J. Diagrams of Kochen-Specker Constructions. *Int. J. Theor. Phys.*, 39:921–926, 2000.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399