THE LOGIC IN COMPUTER SCIENCE COLUMN
by
Yuri GUREVICH

Pairwise Testing*

Andreas Blass† Yuri Gurevich‡

Abstract

We discuss the following problem, which arises in software testing. Given some independent parameters (of a program to be tested), each having a certain finite set of possible values, we intend to test the program by running it several times. For each test, we give the parameters some (intelligently chosen) values. We want to ensure that for each pair of distinct parameters, every pair of possible values is used in at least one of the tests. And we want to do this with as few tests as possible.

1 Introduction

Quisani: I suppose that, with Andreas visiting Microsoft for an extended period, you've been working together on logic and abstract state machines.

*Bull. Eur. Assoc. Theor. Comput. Sci., Oct. 2002

†Partially supported by NSF grant DMS–0070723 and by a grant from Microsoft Research. Address: Mathematics Department, University of Michigan, Ann Arbor, MI 48109–1109, U.S.A., ablass@umich.edu. This paper was written during a visit to Microsoft Research.

‡Microsoft Research, One Microsoft Way, Redmond, WA 98052, U.S.A., gurevich@microsoft.com

Authors: Actually, we've recently been looking into software testing and related combinatorial problems.

Q: Is that connected with your work on abstract state machines?

A: Indirectly. It is related to what our group, the Foundation of Software Engineering group at Microsoft Research, is doing. The group has developed AsmL, a specification language based on abstract state machines. To make AsmL specifications more useful, the group is working on automated derivation of test suites from AsmL specifications. This work generated considerable interest among Microsoft test architects and test engineers.

Q: What combinatorial problems arise in testing?

A: The problem we've looked into is how to generate a test suite (a set of tests) that is *pairwise adequate* in the following sense. Suppose a program involves c parameters, each with a certain number of possible values; say the i^{th} parameter has n_i possible values. A *test* is determined by giving a value for each parameter. We want to find a test suite such that, for each pair of distinct parameters, say the i^{th} and j^{th}, and for each pair of possible values of these parameters, say x and y, there is a test in our suite where the i^{th} parameter has value x and the j^{th} has value y. The objective is to accomplish this with as few tests as possible.

Q: I guess you assume that, for any two parameters, every pair of possible values is relevant.

A: Indeed we do. Moreover, we assume[1] that every assignment of values to the parameters is legal and so can be tested.

Q: Since this is for computer purposes, not for pure mathematics, I assume all the n_i are finite.

A: Of course.

Q: But won't some of them be awfully large? The parameters could, for example, represent floating-point numbers.

A: Yes, but in that case one reduces the number of values by taking into account the program to be tested. The program usually suggests a splitting of the range of a floating-point parameter into a few sub-ranges, within each of

[1]The purpose of this assumption is to simplify our exposition, not to limit the applicability of pairwise testing. It may, for example, be useful to test illegal situations, to see what error messages occur.

which the program acts similarly. Then for testing purposes one uses just one representative (or a small number of representatives) from each sub-range. In the applications that our group has encountered, c and all the n_i were no larger than 50. And often some of the parameters were Boolean ones, so some of the n_i were only 2.

Q: Were you able to make good progress?

A: We did but can't claim much credit. When we began looking at these matters we were told that [8] had the essential information. That paper cites [3, 4] where related ideas are used. We looked at the methods proposed in these papers and felt that certain combinatorial ideas could be profitably applied, giving test suites that have few tests and are easy to find. We developed three such ideas, which we call the "affine," "recursive," and "Boolean" ideas. We realized that some classical combinatorial issues, like Latin squares and projective geometry, are relevant. In the meantime, we traced references back to earlier sources and we also found one recent relevant paper. The affine idea, it turns out, goes back at least to [2] in pure combinatorics, and recently Williams [13] had applied it and the recursive idea to testing. The Boolean idea uses well-known combinatorial results from [5, 6, 7, 12]; their relevance to testing is described in the introduction of [11].

Q: I'd like to hear more about it, and I hope that the record of our discussion may benefit others as well. It is useful to have in one place all those seemingly unrelated facts scattered in the literature as well as your own insight.

A: All right. Let's begin by looking at the methods proposed in the papers we first looked at, [3, 4, 8].

2 Incremental Constructions

Q: Before that, just to make sure I understand the problem, let me check that some things that look obvious to me are really correct.

If you increase the number of parameters (leaving the n_i for the old parameters unchanged), or if you increase some n_i's then the number of tests you need will increase or, at best, stay the same. Also, the number of tests you need is at least the product of the two largest n_i's. If you're so lucky as to have only two parameters ($c = 2$), then this product is exactly the number of tests you need.

A: Right; all these things are indeed obvious. In fact, the product of the two largest n_i's is exactly the number of tests you need even if there are three parameters. But that's not quite so obvious; it will become clear in a while.

Everyone seems to agree that a convenient way to represent a test suite is as a matrix. There is one row for each test, and there is one column for each parameter. The entry in any particular row and column is the value of the column's parameter in the row's test. Pairwise adequacy of the test suite means that the columns are pairwise independent, i.e., for every two columns and every choice of a possible entry in each of them, there is a row where the chosen entries occur in those two columns. (The idea behind the terminology "independent" is that, if you know an entry in one column, that gives you no information about the entry in the same row and another column. To avoid confusion, we stress that we do not ask for probabilistic independence of two columns, where each pair of values would occur in the same number of rows. The simpler sort of independence that we use is sometimes called *qualitative independence*, a terminology that goes back to [9]. We'll omit "qualitative" because we won't need any other sort of independence.)

We'll use the notation r for the number of rows in our matrices (to go with c for the number of columns); thus r represents the number of tests in a suite, and it is what we want to minimize (for fixed c and n_i's).

Let's use the word *requirement* for a set of two ordered pairs, which we write in the form $\{i \mapsto x, j \mapsto y\}$ where i and j are distinct columns (representing parameters) and x and y are possible values of those parameters. We'll say that a matrix *satisfies* this requirement if it has a row in which the entries in columns i and j are x and y, respectively. So we seek matrices that satisfy all the requirements.

With this terminology, we can describe the two methods that we were aware of when we began our work. They both involve building a matrix gradually, trying to satisfy the requirements as efficiently as possible.

2.1 The AETG Method

A: One method, presented in [3, 4], builds the matrix one row at a time. It keeps track of the requirements that remain to be satisfied, and it chooses each new row by a combination of randomization and greed, with a view to satisfying many of the as yet unsatisfied requirements. In more detail, the algorithm creates a new row as follows, as long as any requirements remain

unsatisfied. Choose a column i and a value x for the corresponding parameter such that $i \mapsto x$ occurs in the largest number of unsatisfied requirements. Fill in x as the value in the new row and column i. Then fill in the remaining places in the new row in a random order, choosing the value for each entry in turn so as to maximize the number of previously unsatisfied requirements that become satisfied. Of course, the result here depends on the random ordering. So the authors suggest in [4] that each row be computed some number of times (they suggest 50), with different random orders, and that the best result be used (where "best" means satisfying the most previously unsatisfied requirements).

2.2 The Lei-Tai Method

A: Another method, presented in [8], involves building up the number of parameters gradually, so that at a typical stage in the process one has a test suite for the first c' parameters, where $c' < c$, and one wants to enlarge it to a test suite for the first $c' + 1$ parameters. The method starts with the first two parameters, where, as you observed, it's trivial to find an optimal test matrix. The general step, from c' to $c' + 1$, involves adding to the matrix a new column (for the new parameter) and, unless we're very lucky, adding some new rows to make this column independent of the previous ones. The step proceeds in two parts, a "horizontal" part, filling in the part of the new column that involves existing rows, and a "vertical" part, adding new rows. The authors of [8] observe that, once the horizontal part is completed, there is no choice about the number of new rows to be added in the vertical part. Here are the details. For each value v of the new parameter and for each old column i, let $D(v, i)$, the *deficit* of value v with respect to column i, be the number of unsatisfied requirements of the form $\{i \mapsto x, c' + 1 \mapsto v\}$. At the end of the horizontal part of the step, satisfying such requirements will demand $D(v, i)$ new rows with v in the new column. A single new row can handle requirements from any or all of the old columns i, so the vertical part needs to add

$$D(v) = \max_{i \leq c'} D(v, i)$$

new rows with v in the last column, for a total of

$$\sum_v D(v) = \sum_v \max_i D(v, i)$$

new rows at this step. We'll refer to $D(v)$ as the *(total) deficit* of the value v.

Q: So the real work in this method is in the horizontal part of each step.

A: Essentially. There is some freedom in the vertical part, even though the number of new rows is fixed. For example, which pairs of requirements, involving the same v and different i's, should be satisfied by the same row? And if $D(v,i) < D(v)$ for certain v and i, then there will be some places in column i where any value could be entered. These sorts of freedom might be used to make future steps work better, but this issue is not addressed in [8]. So, indeed, the work is done in the horizontal part.

In fact, two approaches to the horizontal part are proposed in [8]. One approach tries all possible ways to fill in the horizontal part and chooses one to minimize the number of new rows in the vertical part.

Q: Wait a minute. Aren't there usually a huge number of ways to fill in the horizontal part? Will this computation terminate in reasonable time?

A: You're right, and this problem is recognized in [8]. The authors point out that the work can be reduced somewhat, since permuting the values of the new parameter has no effect. Nevertheless, this approach is feasible only for very small cases. That's why a second approach is offered.

The second approach to the horizontal part amounts to the following greedy algorithm. Fill in the entries in the new column one by one, choosing each value so as to maximize the number of previously unsatisfied requirements that become satisfied.

Q: For the first entry to be filled in, any one of the new values will satisfy c' new requirements, so it doesn't matter what value is used for this entry.

A: Right. In fact, for the first few entries in the new column, the greedy strategy can fill in distinct values of the new parameter, as each will satisfy the maximum possible number c' of previously unsatisfied requirements. Things become non-trivial only in rows that remain after each possible value has been used once.

Q: Are the entries in the new column and old rows filled in a random order?

A: The algorithm in [8] just fills them in top-to-bottom order. But one could randomize, try it several times, and take the best of the results.

2.3 A Variation on the Lei-Tai Heuristics

A: There is another way to improve the algorithm. As it stands, the horizontal part is greedily maximizing the number of requirements that get satisfied. But that goal isn't really what's wanted. Suppose a value v has, at the end of the horizontal part, just two unsatisfied requirements and they involve the same column i. Then v will have to be the new-column entry in two new rows, with different elements in column i to satisfy the two requirements. Suppose, on the other hand, that v has, at the end of the horizontal part, more than two unsatisfied requirements, but all referring to different columns. Then these requirements can be satisfied with just one new row having v in the last column. So we are better off in the second scenario, even though the number of unsatisfied requirements is greater. To put it another way, algorithm seeks to minimize

$$\sum_v \sum_i D(v,i)$$

but the number of new rows needed in the vertical part is, as we saw,

$$\sum_v \max_i D(v,i).$$

So we should try to minimize that instead.

We therefore considered modifying the horizontal part of the Lei-Tai algorithm from [8] to greedily reduce this sum of maxima. At each row, put into the new column a value whose $D(v)$ is thereby reduced.[2]

We experimented on some specific values of c and the n_i's, and we found that this modification needs to be refined in order to produce real benefits. The modification is useful in the late stages of the horizontal part, but it seems counterproductive in the early stages. We tried several versions of the algorithm, greedily reducing various weighted combinations of the sum of all the $D(v,i)$'s (as in the Lei-Tai algorithm) and the sum over v of the maximum over i of $D(v,i)$ (as in our proposed modification). The weight factors depended on the row number, giving the Lei-Tai version more weight for the early rows and giving our modification more weight for the later

[2]If there is no such value, then you may want to choose a value v so as to reduce as much as possible the number of i's for which $D(v,i) = D(v)$; the point of that reduction is that it improves our chances of reducing $D(v)$ later.

rows. In our experiments, the best results occurred when the ratio of the two weights was proportional to the cube of the row number. But the number of experiments we did was too limited to support any general claims.

Q: Is this modification of the Lei-Tai algorithm one of the ideas anticipated in [2], [11], and [13]?

A: No. The ideas they anticipated are what we call the affine idea,[3] the Boolean idea, and the recursive idea.

3 The Affine Idea

Q: What are those ideas?

A: Let's start with the affine idea in a special case. Let p be a prime number. Then there is a matrix with p^2 rows, p columns, entries in $\{0, 1, \ldots, p-1\}$, and all columns independent.

Q: So this would correspond to p parameters, each with p values. One of my trivial observations was that we'll need p^2 rows even for just two p-valued parameters, so you're saying it costs no more rows to handle p such parameters than to handle just two.

A: Right. In fact, the method extends to handle $p+1$ such parameters, but let's look at p of them first.

The idea is to work with affine functions over the prime field $\mathbb{Z}/p$ of p elements. Imagine the p^2 rows of our matrix as being labeled with the ordered pairs (a, b) of elements from $\mathbb{Z}/p$ and imagine the p columns as labeled by the elements of $\mathbb{Z}/p$. Then the entry in row (a, b) and column i is $ai + b$, where addition and multiplication are done modulo p, i.e., done in the field $\mathbb{Z}/p$.

Q: So the row labels (a, b) are viewed as designating the p^2 affine functions of one variable over $\mathbb{Z}/p$.

A: Right; that's why we call this construction the affine idea. Figure 1 shows the matrix for $p = 3$. The actual matrix is the part to the right of the vertical line; the column to the left just shows the row labels.

[3] The anticipation covers the affine idea over fields, not the most general case described below.

Figure 1: Affine Matrix for $p = 3$

$0i + 0$	0	0	0
$0i + 1$	1	1	1
$0i + 2$	2	2	2
$1i + 0$	0	1	2
$1i + 1$	1	2	0
$1i + 2$	2	0	1
$2i + 0$	0	2	1
$2i + 1$	1	0	2
$2i + 2$	2	1	0

Q: You don't show the column labels?

A: They're the entries in the row $1i + 0$, so it didn't seem worthwhile to repeat them across the top of the figure.

Q: OK. In this example it's easy to see that the columns are pairwise independent.

A: It's easy to see it in general. Indeed, to satisfy the requirement $\{i \mapsto x, j \mapsto y\}$, we need a row (a, b) such that

$$\begin{aligned} ai + b &= x \\ aj + b &= y. \end{aligned}$$

The determinant of this linear system of equations for a, b is $\begin{vmatrix} i & 1 \\ j & 1 \end{vmatrix} = i - j \neq 0$. As $\mathbb{Z}/p$ is a field, the required a and b exist.

Q: This establishes your claim about p parameters with p values being handled in p^2 tests. You also claimed that this can be improved to handle one more parameter; how do you do that?

A: Add another column, whose entry in row (a, b) is a. See Figure 2. To see that this column is independent of the previous ones, we need to solve for a, b equations of the form

$$\begin{aligned} ai + b &= x \\ a &= y, \end{aligned}$$

Figure 2: Extended Affine Matrix for $p = 3$

$0i + 0$	0	0	0	0
$0i + 1$	1	1	1	0
$0i + 2$	2	2	2	0
$1i + 0$	0	1	2	1
$1i + 1$	1	2	0	1
$1i + 2$	2	0	1	1
$2i + 0$	0	2	1	2
$2i + 1$	1	0	2	2
$2i + 2$	2	1	0	2

and this is trivial. (It does not need that p is prime.)

Q: The obvious question now is whether yet another column can be added, maintaining pairwise independence, without adding more rows.

A: The answer is negative; $p + 1$ is the best you can do. Let's postpone the proof for a while and continue discussing the affine method.

Q: OK. The method looks pretty good if

- the two largest n_i are equal,
- their common value n is prime, and
- $c \leq n + 1$.

Then you get n^2 rows, and we know there's no possibility of doing better. But what if those conditions aren't all satisfied?

A: If the two largest n_i's are not equal but not very different, the affine method may still be reasonably good if the largest n_i is prime and c is at most 1 larger. If the largest n_i is significantly larger than all the others, then a reasonable approach is to first solve the problem without the largest n_i (hence with c reduced by 1) and then to restore the omitted n_i by (the variant described above of) the method of Lei and Tai.

Q: What if the two largest n_i's are equal, say to n, and $c \leq n + 1$, but n isn't prime?

A: A natural approach is to use the affine method with p being the first prime $\geq n$.

Q: How much bigger than n is that p likely to be? I remember reading somewhere that there's always a prime between n and $2n$, but if p is near $2n$ then the number of rows you get, p^2, is unpleasantly large compared to the n^2 that one might hope for.

A: The situation isn't quite that bad. In the first place, the prime number theorem says that the density of primes near n is close to $1/\ln n$, so usually one needs to look only logarithmically[4] past n to find a prime.

Q: That's "usually"; what about the worst case?

A: For that, one needs an expert on the distribution of primes, so we asked Hugh Montgomery of the University of Michigan. He provided most of the following asymptotic information.

There is a constant $\theta < 1$ such that, for all sufficiently large n, there is a prime between n and $n + n^\theta$. The best (smallest) θ currently known is 0.525 (see [1]), but it is conjectured that every positive θ will work. On the other hand, there are, for any K, infinitely many n with no prime between n and $n + K \log n$. There is no consensus (let alone a theorem) about, say, $n + K(\log n)^2$.

So for large n, there is relatively little penalty for going to the next prime: $n^2 + O(n^{1.525})$ tests in the worst case and $n^2 + O(n \log n)$ on average, compared to an optimum n^2 (when the two largest n_i are both n).

In practice, the values of n that arise are not so large. In fact, they are small enough so that a simple lookup table can give the least prime $p \geq n$. For $n \leq 100$ we always have $p \leq n + 7$ (and $p \leq n + 5$ except when $n = 90$ or 91). In addition, the following variants of the affine idea may help.

The affine idea works over any finite field, whether or not it has the form $\mathbb{Z}/p$. Every power q of any prime p is the number of elements of a finite field. For any such q, the affine idea produces a matrix with q^2 rows, $q+1$ columns, q different entries, and all columns independent.

Consider, for example, the 4-element field. It has characteristic 2, i.e., $x + x = 0$ for all x in this field. The four elements are 0, 1, g, and $g+1$ where $g^2 + g + 1 = 0$. The two equations $x + x = 0$ and $g^2 + g + 1 = 0$ (together with the field axioms) determine the arithmetic of this field and thus let us

[4] We use ln for the natural logarithm, base e, and log for the logarithm with base 2.

Figure 3: Extended Affine Matrix over 4-Element Field

$0i+0$	0	0	0	0	0
$0i+1$	1	1	1	1	0
$0i+g$	g	g	g	g	0
$0i+g+1$	$g+1$	$g+1$	$g+1$	$g+1$	0
$1i+0$	0	1	g	$g+1$	1
$1i+1$	1	0	$g+1$	g	1
$1i+g$	g	$g+1$	0	1	1
$1i+g+1$	$g+1$	g	1	0	1
$gi+0$	0	g	$g+1$	1	g
$gi+1$	1	$g+1$	g	0	g
$gi+g$	g	0	1	$g+1$	g
$gi+g+1$	$g+1$	1	0	g	g
$(g+1)i+0$	0	$g+1$	1	g	$g+1$
$(g+1)i+1$	1	g	0	$g+1$	$g+1$
$(g+1)i+g$	g	1	$g+1$	0	$g+1$
$(g+1)i+g+1$	$g+1$	0	g	1	$g+1$

write down a 16 by 5 matrix (Figure 3) with 4 entries and with all columns independent.

Q: So in effect a prime power is as good as a prime.

A: Right, provided you don't mind doing arithmetic in general finite fields.

There is another variant of the affine method that uses rings of the form $\mathbb{Z}/q$ for non-prime values of q.

Q: Those rings aren't fields, so your proof that the columns are independent won't work.

A: Right; in fact the columns won't all be independent, but some of them will be. Specifically, if we let i range from 0 to $p-1$ where p is the smallest prime divisor of q, then the columns labeled i will be pairwise independent. Indeed, proceeding as in the field case, we find that when i and j are in this range and distinct then the equations we must solve for a and b have a determinant $i-j$, which is non-zero and smaller than p in absolute value. Thus, this determinant is relatively prime to q and therefore invertible in $\mathbb{Z}/q$. Therefore, the required a and b exist (and are unique).

Q: So you get a matrix with q^2 rows, q entries per column, pairwise independent columns, but only p columns, rather than the q columns you could get using a field.

A: Right. You can get one more column as before; its entry in row (a, b) is a.

Q: Could you get more columns by using as your column labels not simply 0 through $p-1$ but some cleverly chosen set of more than p elements of $\mathbb{Z}/q$?

A: No. No matter how clever you are, any set of more than p elements will have two members that are congruent modulo p. Then the two corresponding columns will not be independent.

Q: So $p+1$ is the best you can hope for with the affine method over $\mathbb{Z}/q$. That looks considerably worse than what you get by using finite fields.

A: This method is useful if the required number c of columns happens to be small compared to the largest n_i. Even if the given c is large, the recursive idea (to be explained in a moment) sometimes leads to sub-problems with small c.

4 The Recursive Idea

A: Let n be the largest n_i, and suppose that c is significantly larger than n. The affine idea requires us to use a prime power q that is at least $c-1$ (or, in the ring version, a number whose smallest prime divisor is at least $c-1$, which is even worse). So we get a matrix with $q^2 \geq (c-1)^2$ rows. Every column of the matrix has q distinct entries, far more than we need, but the affine method can't take advantage of this to reduce the number of rows.

The affine method can, nevertheless, make a contribution even in the case of large c. One approach is to begin by considering only some subset of the parameters, small enough so that the affine method provides a good matrix, and then to extend this matrix by the Lei-Tai method, perhaps modified as discussed above, to incorporate the rest of the parameters.

Q: So you'd use a matrix built using the affine idea instead of the two-column matrix that Lei and Tai use as the starting point for their construction.

A: Right. We experimented a bit with this method on one of the examples given by Lei and Tai, 20 parameters with 10 values each. We used the affine

Figure 4: Partial Matrix for 12 Three-Valued Variables

0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	0
2	2	2	0	2	2	2	0	2	2	2	0
0	1	2	1	0	1	2	1	0	1	2	1
1	2	0	1	1	2	0	1	1	2	0	1
2	0	1	1	2	0	1	1	2	0	1	1
0	2	1	2	0	2	1	2	0	2	1	2
1	0	2	2	1	0	2	2	1	0	2	2
2	1	0	2	2	1	0	2	2	1	0	2

method with $p = 11$. The results were somewhat better, but not dramatically better, than the Lei-Tai method.

4.1 Multiple Layers

A: There is another, more systematic way to use the affine method with large c. We call it the recursive idea. Choose $q \geq n$ as before but $q < c$. Let w be the number of columns we can handle with this q, namely $q + 1$ if q is a prime power (and we use the field of size q) or $p + 1$ where p is the smallest prime divisor of q (if we use $\mathbb{Z}/q$). Form the $q^2 \times w$ matrix given by the affine method, and repeat it horizontally to fill the required c columns. Columns i and j are independent except when i and j are congruent modulo w. Figure 4 shows an example where $n = 3$ and $c = 12$. (The vertical lines in the figure are just to make the copies of the original, 4-column matrix easier to see.)

Now we need to add more rows to make the columns within each congruence class modulo w pairwise independent. There are w congruence classes, each of size c/w (rounded to an integer). For each congruence class, we handle the columns in that class separately, by adding copies of the original $q^2 \times w$ matrix below the matrix already constructed. In our example, the first congruence class is handled as shown in Figure 5.

We'll refer to the newly added rows as the second layer of the matrix (and the original rows as the first layer).

Figure 5: More of the Matrix for $c = 12$ and $n = 3$

0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	0
2	2	2	0	2	2	2	0	2	2	2	0
0	1	2	1	0	1	2	1	0	1	2	1
1	2	0	1	1	2	0	1	1	2	0	1
2	0	1	1	2	0	1	1	2	0	1	1
0	2	1	2	0	2	1	2	0	2	1	2
1	0	2	2	1	0	2	2	1	0	2	2
2	1	0	2	2	1	0	2	2	1	0	2
0				0				0			
1				1				1			
2				2				2			
0				1				2			
1				2				0			
2				0				1			
0				2				1			
1				0				2			
2				1				0			

Figure 6: The Whole Matrix for $c = 12$ and $n = 3$

0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	0
2	2	2	0	2	2	2	0	2	2	2	0
0	1	2	1	0	1	2	1	0	1	2	1
1	2	0	1	1	2	0	1	1	2	0	1
2	0	1	1	2	0	1	1	2	0	1	1
0	2	1	2	0	2	1	2	0	2	1	2
1	0	2	2	1	0	2	2	1	0	2	2
2	1	0	2	2	1	0	2	2	1	0	2
0	0	0	0	1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2	0	0	0	0
2	2	2	2	0	0	0	0	1	1	1	1
0	0	0	0	2	2	2	2	1	1	1	1
1	1	1	1	0	0	0	0	2	2	2	2
2	2	2	2	1	1	1	1	0	0	0	0

Q: The first three rows in the second layer of Figure 5 don't contribute to the independence of these columns. They just duplicate earlier rows.

A: That's right, so we can omit these three rows. Notice, though, that if we had $c \geq 13$ and thus had four columns in some equivalence class, then only the first row of the second layer would be redundant. The second and third rows would not be constant, so they actually contribute to the independence.

In our example, omitting the three redundant rows and handling the other congruence classes similarly, we get Figure 6.

That finishes the job in this example, but in general two columns would still be identical if their positions differed by a multiple of w^2. So if c were greater than w^2 then we'd need a third layer to handle these congruence classes. In general, we get a matrix with $\lceil \log_w c \rceil = \lceil \log c / \log w \rceil$ layers of $q^2 - 1$ rows each, except that the first layer has q^2 rows. (The $q^2 - 1$ comes from the fact that the first, all-zeros row would be the same in every layer, so it can be omitted from all layers but one.)

Q: It seems the extra column added to the affine method, the column not given by affine functions, is causing trouble here by allowing only the first

row of each layer to be omitted.

A: That's true. If we used the original, "pure," affine method, then each layer but the first would have only $q^2 - q$ rows, because all the affine functions of the form $0i + b$ would give redundant rows. On the other hand, w would be decreased by 1, so there is a danger that we might need an additional layer. So there is a trade-off, whose value often depends on the effect of rounding the ratio of logarithms to an integer.

One can improve on this pure affine method as follows. In the first layer, use the affine method with the extra column. Any two columns that are not yet independent are in fact identical. So in subsequent layers we don't need any constant rows. So in all layers but the first we can use the pure affine method minus its first q rows.

Q: How is this better than the pure affine method? They both use q^2 rows in the first layer and $q^2 - q$ rows in each subsequent layer.

A: Yes, but the improvement uses a larger w in the top layer, so the congruence classes to be treated in subsequent layers are a little smaller. As a result, we may get by with one fewer layer.

There is another variation that is occasionally useful. We can choose q independently for the different layers. This may be particularly useful at the last layer, where the congruence classes remaining to be treated may be smaller than n and we can therefore use a q that is not a prime power and still have w large enough.

Q: You mentioned that the affine and recursive ideas are already in the literature.

A: Yes, Bose [2] uses the affine idea (over fields, not over general $\mathbb{Z}/q$) to construct what amount to test matrices, though he uses the terminology of Latin squares. (When he wrote his paper, software testing and even computers were far in the future.) Williams [13] applies Bose's construction along with the recursive idea to the problem of testing.

4.2 An Explicit Formula

A: It seems useful to observe — and it doesn't seem to be in the literature — that the matrices given by the pure affine idea for a prime p plus the recursive idea admit a rather simple, explicit description. Number the rows

and columns starting at 0. The entry in row z and column i is z if $0 \le z < p$ and

$$\left(a \left\lfloor \frac{i}{p^l} \right\rfloor + b \right) \bmod p, \text{ if } z = p + l(p^2 - p) + (a-1)p + b,$$

with $1 \le a < p$, $0 \le b < p$, and $0 \le l$.

Q: I'm not sure that's really "rather simple". I understand "z if $0 \le z < p$", which just means that the first p rows are constant. But please explain the non-trivial part of the formula.

A: OK, let's look at the formula piece by piece, starting with the part $z = p + l(p^2 - p) + (a-1)p + b$. This is just a compact way of saying that row z is in layer l, where we start numbering layers at 0, and within layer l it is row number $(a-1)p + b$. Remember that, when we use the pure affine method, each layer but the first has $p^2 - p$ rows, so $p + l(p^2 - p)$ is the number of rows in layers strictly preceding layer l. (For uniformity of terminology here, we regard the first p rows, the constant ones, as strictly preceding layer 0.) So, as we start numbering rows at 0, layer l begins at row $p + l(p^2 - p)$ and continues to row $p + (l+1)(p^2 - p) - 1$, inclusive. This is precisely the range of $p + l(p^2 - p) + (a-1)p + b$ as a ranges from 1 to $p - 1$ and b ranges from 0 to $p - 1$.

Now within layer l, increasing z corresponds to lexicographically increasing the pair (a, b). This is because b ranges only from 0 to $p - 1$, so any increase of a, even by only 1, increases $(a-1)p + b$ by at least p and thus outweighs any decrease in b. So as z increases through layer l, the pair (a, b) runs through all the relevant coefficients for affine functions $ax + b$ over $\mathbb{Z}/p$, in lexicographic order. (Remember that $a = 0$ is omitted since it would produce constant rows, duplicating rows we already had earlier.)

It remains to see how these affine functions are used. Let's look first at the easiest case, layer 0. Here we should have, in the first p columns, the matrix given by the affine method, and in subsequent columns simply repetitions of this matrix. So in the first p columns, where $0 \le i < p$, we should have entries $ai + b$, calculated in $\mathbb{Z}/p$, i.e., calculated modulo p. These entries and the repetitions in later columns, with period p, are given by $a(i \bmod p) + b$ calculated modulo p, which simplifies to $ai + b \bmod p$. (It's a simplification of the mathematical description, reducing modulo p only once, at the end. For computation, it may be better to reduce as you go rather than waiting until the end.) And this simplified formula is exactly the one we had above, when $l = 0$, since then $\lfloor i/p^l \rfloor = i$.

In the next layer, where $l = 1$, we take care of the pairs of columns that have not already achieved independence, namely those whose numbers are congruent modulo p. And we take care of them by putting more copies of the same affine matrix, but with columns labeled differently. Thus, the first p columns ($0 \le i < p$) are each the first in their respective congruence classes, so their entries in layer 1 are like the entries of the first column, $i = 0$, in layer 0. Similarly, the next p columns ($p \le i < 2p$) get, in layer 1, the entries that the next single column, $i = 1$, had in layer 0. Continuing in this way, we see that columns from pj to $(p+1)j - 1$ inclusive get, in layer 1, the entries that column j had in layer 0. In other words, the layer 1 entries in column i are the layer 0 entries in column $j = \lfloor i/p \rfloor$, namely

$$\left(a \left\lfloor \frac{i}{p} \right\rfloor + b \right) \bmod p.$$

Q: OK. I see how, when you continue recursively, you'll get higher powers of p in the denominator, because in layer l you're concerned with pairs of columns whose numbers i are congruent modulo p^l. Now that I see what's going on, the formula does look simple.

5 The Boolean Idea

Q: What is the third idea you mentioned, the Boolean idea?

A: As the name might suggest, it applies in the special case where all the parameters are Boolean, i.e., all $n_i = 2$. Then a solution of the testing problem, an $r \times c$ Boolean matrix, can be viewed as a family (indexed by the columns, i.e., by the parameters) of subsets of an r-element set, say $\{1, 2, \ldots, r\}$. The set indexed by i consists of those m such that the entry in row m, column i is `true` (or 1).

Independence of the rows is equivalent to the following requirements on these sets:

- No one is a subset of another.
- No two are disjoint.
- No two cover the whole set $\{1, 2, \ldots, r\}$.

Indeed, the first of these three requirements says that the pairs of values $(1,0)$ and $(0,1)$ occur in each pair of columns, the second requirement says that $(1,1)$ occurs, and the last requirement says that $(0,0)$ occurs.

An equivalent way of expressing these requirements is:

- No two of the sets are complementary.
- The sets and their complements form an antichain (i.e., none is included in another).

So we are looking for antichains, closed under complementation, in the Boolean algebra of subsets of $\{1,2,\ldots,r\}$. Then our test matrix is formed by choosing, in this antichain, one member of each complementary pair and using (the characteristic functions of) the chosen sets as the columns of our matrix.

How big can a complement-closed antichain of subsets of $\{1,2,\ldots,r\}$ be? Half of that will be the largest c that can be handled with r rows in the Boolean situation.

If r is even, Sperner's theorem [12] says that the largest possible antichain of subsets of an r-set consists of the subsets of size $r/2$. Since this antichain is closed under complementation, we find that the largest c that can be handled with r rows is

$$c = \frac{1}{2}\binom{r}{r/2} = \binom{r-1}{r/2}.$$

If r is odd, Sperner's theorem says that the largest antichains have size $\binom{r}{(r-1)/2}$, but these are not closed under complementation.

You can construct a complement-closed antichain by fixing one element in the r-set, say 1, and taking all of the $(r-1)/2$-element subsets that contain 1 and all of the $(r+1)/2$-element subsets that don't contain 1.

It turns out (as a consequence of the Erdős-Ko-Rado theorem [5]) that these are the largest complement-closed antichains. So the largest c that can be handled with r rows is

$$c = \binom{r-1}{(r-3)/2}.$$

Q: What is this Erdős-Ko-Rado theorem?

A: The theorem asserts that, for any r and any $k \leq r/2$, if $\mathcal{F}$ is a family of k-element subsets of an r-set and if every two elements of $\mathcal{F}$ intersect, then the cardinality $|\mathcal{F}|$ is at most $\binom{r-1}{k-1}$.

Q: So it's not directly about complement-closed antichains. How does it imply a bound on the size of those?

A: Well, suppose $\mathcal{A}$ is a complement-closed antichain of subsets of an r-set where r is odd. Let $\mathcal{B}$ consist of those elements of $\mathcal{A}$ whose cardinality is smaller than $r/2$. Notice that the elements of $\mathcal{A}$ that are not in $\mathcal{B}$, namely those of size greater than $r/2$, are exactly the complements of the sets in $\mathcal{B}$. Notice also that every two sets in $\mathcal{B}$ intersect, for if X were disjoint from Y then X would be a subset of the complement of Y, contradicting the fact that $\mathcal{A}$ is an antichain.

Q: So if all the sets in $\mathcal{B}$ had the same size $k < r/2$, then the Erdős-Ko-Rado theorem would imply that

$$|\mathcal{B}| \leq \binom{r-1}{k-1} \leq \binom{r-1}{(r-3)/2},$$

which agrees with your claim. But I don't see why all the sets in $\mathcal{B}$ should have the same size.

A: They won't in general, but we can modify $\mathcal{B}$, without decreasing its cardinality, and without losing its crucial properties that it is an antichain and that every two members meet, yet so that after the modification all its members have size $(r-1)/2$.

Q: How do you do that? If it works, it clearly finishes the proof.

A: We'll need the following well known fact.

Lemma 1 *If $k < r/2$ then there is a one-to-one function that assigns to each k-element subset of an r-set a $(k+1)$-element superset of it.*

Q: I haven't seen this lemma before, but I think I see how it will finish your proof. Apply the lemma to the smallest elements in $\mathcal{B}$, and replace each of these smallest elements of $\mathcal{B}$ by its image, a set larger by 1. This clearly preserves the property that all pairs of sets from $\mathcal{B}$ meet, as we've replaced certain sets by supersets. It also preserves the antichain property, equally

trivially. And it doesn't decrease the size of $\mathcal{B}$ because the newly added sets were not there before, as $\mathcal{B}$ was an antichain.

So, as long as the minimum size of sets in $\mathcal{B}$ is smaller than $(r-1)/2$, we can increase it. Repeating this process, we end up with all sets in $\mathcal{B}$ having size $(r-1)/2$, so the Erdős-Ko-Rado theorem applies.

But how do you prove the lemma?

A: It's an application of the matching theorem. By that theorem, it suffices to show that, if we take any collection $\mathcal{X}$ of k-element subsets of our r-set and if we define

$$\mathcal{Y} = \{Y : |Y| = k+1 \text{ and } \exists X \in \mathcal{X}\ X \subset Y\},$$

then $|\mathcal{X}| \leq |\mathcal{Y}|$. And this is easy if we just count in two ways the number P of pairs (X, Y) with $X \in \mathcal{X}$, $Y \in \mathcal{Y}$ and $X \subset Y$.

On the one hand, each $X \in \mathcal{X}$ occurs in exactly $r-k$ such pairs, for any element of the complement of X can be added to form a Y. So $P = (r-k)|\mathcal{X}|$. On the other hand, any $Y \in \mathcal{Y}$ occurs in at most $k+1$ such pairs, as each possible X is obtained by removing one of the $k+1$ elements of Y. So $P \leq (k+1)|\mathcal{Y}|$. Therefore

$$(r-k)|\mathcal{X}| \leq (k+1)|\mathcal{Y}|.$$

As $k < r/2$, we have $r - k \geq k+1$, and it follows that $|\mathcal{X}| \leq |\mathcal{Y}|$.

Q: So for Boolean parameters, you know exactly how many pairwise independent columns can be obtained from a given number of rows. How does this compare with what the affine and recursive ideas give you?

A: Using Stirling's approximation for the factorials involved in the binomial coefficients, we find that, for large r,

$$c \sim 2^{r-1}\sqrt{\frac{2}{\pi r}}.$$

Inverting this, to express r in terms of c, we get

$$r = \log c + O(\log\log c).$$

The constant in the $O(\dots)$ is small; any constant $> 1/2$ will do.

For comparison, the affine and recursive ideas give, for the Boolean case, asymptotically $2\log c$.

We also note that one of the examples given in [8] for comparison with [3, 4] involved only Boolean parameters, 100 of them. For this example, the method of [8] gives a matrix with 15 rows, and the method of [3, 4] gave a matrix of 12 rows.[5] The Boolean method needs only 10 rows for this example. Furthermore, the matrix is very easy to write down. The first row can be taken to be all 0's. Then in each column fill in exactly five 1's and four 0's in the remaining spaces. Do this in such a way that all the columns are different; for example, go through the 5-element subsets of a 9-element set in lexicographic order. (There are 126 such sets, so we could in fact handle an additional 26 columns without needing any more rows.)

Q: Do the authors of these testing papers know about the Boolean method?

A: Some of them certainly do. The fact that 100 Boolean parameters can be handled with 10 tests is stated in both [3] and [4], and the latter refers to [11] where the general result is stated. But the context in which this 10 occurs in [3, 4] is a comparison of methods for getting independence with methods that get probabilistic independence. It is not suggested that the Boolean method should be used in practice or combined with other known methods.

It may be worth emphasizing that the matrices given by the Boolean method are very easy to construct. Essentially, one just has to lexicographically (or in some other convenient order) generate subsets of a fixed size in a fixed set.

Q: So the case of 2-valued variables works as nicely as one could hope. The exact optimum is known and an optimal matrix is easy to produce. Might there be similar good news for 3-valued variables?

A: Sloane briefly discusses the case of 3-valued variables in the introduction of [11]. Unfortunately the situation is nowhere near as nice as in the 2-valued case.

As we saw above, the optimal r for c 3-valued parameters is 9 for $c = 2, 3, 4$. The optimal r for $c = 5$ is 11. Beyond that, Sloane gives a table of the best known r's for c up to 12, he gives references for some explicit constructions, and he cites an asymptotic result that the optimal r satisfies $r = \frac{3}{2} \log c(1 + o(1))$.

[5] These numbers are taken from [8]. Note that the methods involve non-determinism or randomization. It appears that numerous repetitions are needed to get results this good.

It seems to be a difficult combinatorial problem to say more about the case of 3-valued variables. If you're interested, you should probably look up the work cited in [11].

6 Random Attempts

Q: What would happen if, instead of applying these combinatorial or algebraic methods, you just tried to build a test matrix at random?

A: That's an interesting question. It would be a shame if all this work didn't produce something better than randomness.

Consider the case of c parameters, each with the same number n of possible values. Let's fill in the entries of an $r \times c$ matrix at random, with n possible values for the entries. The questions to consider are:

- What is the probability of getting all the columns independent?
- How big must r be so that this probability is non-zero (so r rows suffice for c columns when all $n_i = n$)?
- How big must r be so that this probability is large (so that we can generate an appropriate matrix at random)?

If r is as in the second question, then we have a probabilistic proof of the existence of an $r \times c$ test matrix for n-valued parameters. If r is as in the third question, then we have a good chance of finding such a test matrix by randomization.

In fact, there's a better way to use randomization. As soon as r is big enough to make the probability p of success not too tiny, we can choose a moderate-sized k and build k/p random $r \times c$ matrices. We'll have a good chance that at least one of them will have pairwise independent columns. The "good chance" depends on k, but $k = 20$ will surely do for practical purposes; so we just need r big enough so that $20/p$ is a feasible number of attempts.

Notice, by the way, that the actual testing, with the necessary repetitions, may involve more computation time than producing the test matrix. So it may be worthwhile to expend considerable computational effort on getting the test matrix to have few rows.

Now let's estimate the probabilities. The probability that a particular requirement $\{i \mapsto x, j \mapsto y\}$ (meaning entries x, y in columns i, j) is satisfied by a particular row is $1/n^2$. So the probability that it is *not* satisfied by *any* row is, since the rows are independent in the probabilistic sense,

$$\left(1 - \frac{1}{n^2}\right)^r.$$

So the expected number of unsatisfied requirements is

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

This is an upper bound for the probability that some requirement is unsatisfied. So when this expectation is < 1 then there exists an $r \times c$ matrix with entries from an n-element set and with all pairs of columns independent.

For the expectation to be < 1, we need approximately

$$r > 2n^2 \ln c + n^2(2 \ln n - \ln 2).$$

Note that this involves a term proportional to $n^2 \log n$, which our methods didn't need. Also, the other main term is

$$2n^2 \ln c = 2n^2 \frac{\log c}{\log e} = \frac{n^2 \log c}{\log \sqrt{e}}.$$

Since $\sqrt{e} < 2$, this term is larger than $n^2 \log c$, whereas the affine and recursive ideas together give asymptotically $n^2 \log c / \log w$.

Q: So randomization is not the best option.

A: Probably not. Notice, though, that we computed an r for which the expected number of unsatisfied requirements is < 1. Since this expectation is an upper bound for the probability that some requirement is unsatisfied, the r we found certainly has weaker property that there is a non-zero probability of satisfying all requirements. But it is imaginable that a smaller r might have this weaker property, which is what we really want.

Finally, to answer the third of our questions above, if we increase r to $2n^2 \ln c + n^2(2 \ln n - \ln 2) + An^2$ then the probability of not getting all columns independent decreases to less than e^{-A}. So by choosing a moderate sized A, we have a good chance of getting a solution to the testing problem by just filling in the matrix at random.

In fact, if n isn't too big, we could choose $A = 1/n^2$, i.e., we could add just one more row to the number r computed above. The probability of not getting pairwise independent columns would be at most e^{-1/n^2}. By repeating the random experiment $O(n^2)$ times, we'd have a good chance of finding a matrix with pairwise independent columns.

7 Projective Planes

Q: You promised to explain why there can't be a $q^2 \times (q+2)$ matrix with q entries and independent columns. In other words, why can't the matrix given by the affine method, which you already extended by one more column, be extended by even more columns without adding rows?

A: The reason for that is best explained in terms of projective planes of order q.

Q: Remind me what a projective plane is.

A: A projective plane consists of a set P of points, a set L of lines, and an incidence relation between them (the relation being expressed by saying that a point lies on a line or that a line goes through a point), subject to three axioms:

- Every two distinct points lie on a unique common line.
- Every two distinct lines go through a unique common point.
- There exist four points of which no three lie on a common line.

A projective plane has order q if there are $q+1$ points on every line. It is a theorem that all lines have the same number of points; also, every point of a projective plane of order q lies on exactly $q+1$ lines.

Q: What does this have to do with test matrices?

A: It turns out that a $q^2 \times (q+1)$ matrix with q entries and pairwise independent columns is essentially the same thing as a projective plane of order q. (We're assuming $q \geq 2$. The matrix for $q = 1$ is trivial and what should be a projective plane of order 1 is (intentionally) excluded by the third axiom for projective planes.)

Q: What does "essentially the same thing" mean?

A: Probably the best way to explain it is to prove it, but for a formal definition you could say that there are two constructions, one converting the matrices in question into projective planes, and one in the reverse direction, such that both composite constructions are the identity up to isomorphism.

Q: That's pretty abstract, and you say that the proof will make it clearer, so please show me the proof.

A: OK. First, suppose we have a matrix of the specified sort. Notice that, in accordance with one of your observations near the beginning of our discussion, in every two distinct columns, every pair of entries will appear together in exactly one row.

Q: Right. Independence requires each pair to appear together, and there are just barely enough rows for that, so no pair can appear together twice.

A: In particular, each of the q entries appears exactly q times in every column.

Now define a projective plane as follows. The points are the $(q+1)q$ pairs (i, x) where i labels a column and x is an entry, plus one special point called $*$. The lines and the incidence relation are defined as follows. First, there is a line for each column; the line for column i is incident with the q points (i, x) and with $*$. Second, there is a line for each row; the line for row m is incident with those points (i, x) such that the entry in row m and column i is x.

Every two distinct points lie on a unique common line. Indeed, if the points are (i, x) and (j, y) with $i \neq j$, then the independence of columns i and j gives a row whose line contains these points, and we saw above that there is only one such row. Furthermore, no column gives a line containing both, as $i \neq j$. If, on the other hand, the points are (i, x) and (i, y), then column i gives the required line and it is clearly unique. Finally, if the points are (i, x) and $*$, then again column i gives the unique line through both.

Every two distinct lines lie on a unique common point. It suffices to prove existence, as uniqueness is equivalent to the uniqueness proved in the preceding paragraph. (Both say that you can't have two distinct lines going through two distinct points.) If the two lines come from columns, then $*$ is a common point. If one line comes from column i and the other from row m, then the common point is (i, x) where x is the entry in row m column i. The non-trivial case is that both of the lines come from rows, say rows m and m'. What we must prove is that there is a column in which these two rows have the same entry.

To this end, we count in two ways the number of pairs $(i, \{s, t\})$ where i is a column, s and t are distinct rows, and these two rows have the same entry in column i. For the first count, we use the fact that each entry occurs exactly q times in each column. So a particular column and a particular entry will contribute $q(q-1)/2$ pairs to our count. As there are $q+1$ columns and q entries, the total count is

$$(q+1) \cdot q \cdot \frac{q(q-1)}{2} = \frac{q^2(q^2-1)}{2}.$$

The second way of counting the pairs is as follows. There are q^2 rows, so there are exactly $q^2(q^2-1)/2$ possible second components $\{s, t\}$ for one of our pairs. Each such $\{s, t\}$ contributes either one pair $(i, \{s, t\})$ or none, depending on whether there is a column i with equal entries in rows s and t. (We saw already that there can't be two such columns.) The only way for the second count to match the first is for *every* possible $\{s, t\}$ to have an appropriate i and so to contribute a pair. This completes the proof of the second axiom of projective planes.

Q: You can skip the third axiom; I already see how it follows immediately from what you've proved and $q \geq 2$.

A: OK. Now we can answer your question about why you couldn't have a matrix like those considered here but with $q+2$ columns.

Suppose you had such a matrix. Pick arbitrarily one column, call it column i, and let s and t be two rows in which column i has the same entry. (Remember that each column has each entry q times and $q \geq 2$.) Now imagine the matrix without column i. That determines a projective plane. In particular (by the hardest part of the proof just completed), there is a column j where rows s and t have the same entry; and $j \neq i$ since we're looking at the matrix without column i. But now look at another submatrix, obtained by restoring column i and deleting some column other than i and j. This has two columns, i and j, in which rows s and t have equal entries, contrary to what we proved above.

Q: That answers my original question, but now I'm curious about something else. You produced a projective plane from a matrix, but you also claimed to be able to go in the other direction. How does that work?

A: We need to use the well-known fact that a projective plane of order q has exactly q^2+q+1 points and the same number of lines. Also, as mentioned

earlier, each line has $q+1$ points on it and each point has $q+1$ lines through it. If you grant these facts, then the construction is easy.

Given a projective plane of order q, choose arbitrarily one of its points and call it $*$. On each of the $q+1$ lines through $*$, arbitrarily label the q points other than $*$ by labels $1, 2, \ldots, q$. Now build a $q^2 \times (q+1)$ matrix as follows. The $q+1$ columns are labeled by the $q+1$ lines l through $*$. The q^2 rows are labeled by the remaining q^2 lines. The entry in row m and column l is the label of the intersection point of lines l and m. To see that the columns are pairwise independent, consider any two distinct columns, say labeled by lines l and l', and any pair of labels x and x' in $\{1, 2, \ldots, q\}$. Let p be the point on l that was labeled x, and let p' be the point on l' that was labeled x'. Then the line m through p and p' does not go through $*$. (Proof: $*$ is the unique intersection point of l and l'. In particular p' is not on l. But p' is on m so $m \neq l$. Since l is the unique line through p and $*$ and since p is on m, it follows that $*$ is not on m.) So m determines a row of our matrix, and this row clearly has entries x and x' in columns l and l', respectively.

Q: So instead of using the affine method, starting with a field, we could get equally good results starting with any projective plane. So maybe we're not limited to prime powers for q.

A: Maybe. It's an open problem whether there are any finite projective planes whose orders are not prime powers. The first two non-prime-powers, 6 and 10, are known not to be orders of projective planes, but the question is open for 12.

Q: So it might be that the only finite projective planes are those arising from finite fields.

A: No. It is known that there are finite projective planes not isomorphic to those given by fields. But all those examples have prime power order. So, in going from the construction using fields to general projective planes, you get new examples, but not (yet) new orders of examples. By the way, the smallest order for which there are non-isomorphic projective planes is 9, and there are exactly 4 isomorphism classes for this order.

8 Latin Squares

Q: You also promised to explain why the product of the two largest n_i's is an adequate number of rows not only in the obvious case where $c = 2$ but

Figure 7: A Latin Square

	0	1	2
0	0	2	1
1	2	1	0
2	1	0	2

also for $c = 3$. And, while we're on that subject, what happens when $c = 4$?

A: The situation for $c = 3$ is probably easiest to see with an alternative way of describing the test matrices. Let's assume, to simplify the notation, that $n_1 \geq n_2 \geq n_3$. So we're looking for a matrix with $n_1 n_2$ rows. If the desired matrix exists, then each pair of values for the first two parameters occurs in exactly one row, so we can label the rows by these pairs of values. To visualize this, imagine an n_1 by n_2 array,[6] the rows (resp. columns) being labeled by the values of the first (resp. second) parameter. So each location in this array corresponds to a row of the original matrix. The third column of the matrix can then be represented by copying its entry from any row of the matrix into the corresponding location of the array. For example, the 3-column matrix in Figure 1 gives rise to the array in Figure 7.

The independence of the third column from each of the others is reflected in the fact that every value for the third parameter occurs in every row and in every column. When $n_1 = n_2 = n_3$ as in this example, such an array is called a *Latin square*.

Now you can see why $n_1 \cdot n_2$ rows in the matrix suffice when $c = 3$. We can fill in the array by putting the n_3 possible entries into the first n_3 spaces in the first row (there's enough room as $n_3 \leq n_2$) and then repeating this pattern in the subsequent rows, cyclically shifting it by one space from each row to the next. Then the remaining spaces in the table can be filled in arbitrarily, and the result will have the required properties. It's obvious that every one of the entries appears in every row. That it also appears in every column is because $n_1 \geq n_2$ so there are enough rows for the cyclic shifts to go all the way around.

Q: So what about four parameters? Is the product of the two largest n_i's

[6] To avoid confusion, we'll consistently use "array" for this representation and "matrix" for the representation used in the preceding sections.

Figure 8: A Graeco-Latin Square

	0	1	2
0	0/0	2/1	1/2
1	2/2	1/0	0/1
2	1/1	0/2	2/0

always a sufficient number of rows to handle four columns?

A: No, not even in the case that all the n_i have the same value n. The first counterexample is $n = 6$.

Notice that, if the matrix has four columns, then we can represent the third column by an array as above and the fourth column by a second such array. Each of the two arrays has, as above, the property that every value of the relevant parameter occurs in every row and in every column. Furthermore, for every pair of values of the third and fourth parameters, there must be a location occupied by those two values in the two arrays. One often superimposes the two arrays, which makes this last condition easier to check. For example, the matrix of Figure 2 gives the array (or superposed pair of arrays) of Figure 8.

When, as in this example, $n_1 = n_2 = n_3 = n_4$, we have two superposed Latin squares with the additional condition that every possible pair of entries occurs (and occurs exactly once, because there isn't enough room for more occurrences). This additional condition is often called *orthogonality*, and a pair of orthogonal Latin squares is often called a *Graeco-Latin square*. (Apparently the values of one parameter were traditionally represented by Greek letters and the values of the other by Latin letters.)

In general, to say that n^2 rows suffice for a matrix with c columns, n entries in every column, and all columns independent is the same as to say that there are $c - 2$ pairwise orthogonal Latin squares of order n.

The question about always getting a fourth column, in the special case where all the n_i are equal, thus comes down to whether Graeco-Latin squares of all orders exist.

Q: So when n is a prime power, then there are, by what we saw earlier, $n-1$ pairwise orthogonal Latin squares of order n, and there do not exist n of them.

A: Right. The first case not settled by the affine idea (or equivalently by projective planes) is $n = 6$.

That there do not exist two orthogonal Latin squares of order 6 is a fairly classical combinatorial result, confirming the first nontrivial case of a conjecture of Euler. (Euler conjectured that there do not exist two orthogonal Latin squares of order n whenever $n \equiv 2 \pmod 4$. The first two cases, $n = 2$ and 6, are correct, but all other cases are wrong.)

References

[1] Roger C. Baker, Glynn Harman, and János Pintz, "The difference between consecutive primes. II," *Proc. London Math. Soc. (3)* 83 (2001) 532–562.

[2] R. C. Bose, "On the application of the properties of the Galois fields to the construction of hyper Graeco-Latin squares," *Sankhya* 3 (1938) 323–338.

[3] David M. Cohen, Siddharta R. Dalal, Jesse Parelius, and Gardner C. Patton, "The combinatorial design approach to automatic test generation," *IEEE Software*, September 1996 83–87.

[4] David M. Cohen, Siddharta R. Dalal, Michael L. Fredman, Gardner C. Patton, "The AETG system: An approach to testing based on combinatorial design," *IEEE Trans. on Software Engineering* 23 (1997) 437–444.

[5] Paul Erdős, Chao Ko, and Richard Rado, "Intersection theorems for systems of finite sets," *Quart. J. Math. Oxford Ser. (2)* 12 (1961) 313–320.

[6] Gyula O. H. Katona, "Two applications (for search theory and truth functions) of Sperner type theorems," in *Collection of articles dedicated to the memory of Alfréd Rényi,II. Period. Math. Hungar.* 3 (1973), 19–26.

[7] Daniel J. Kleitman and Joel Spencer, "Families of k-independent sets," *Discrete Math.* 6 (1973) 255–262.

[8] Yu Lei and Kuo-Chung Tai, "A test generation strategy for pairwise testing," *Proc. 3rd IEEE High-Assurance Systems Engineering Symposium*, November 1998.

[9] Edward Marczewski, "Indépendance d'ensembles et prolongement de mesures (résultats et problèmes)," *Colloquium Math.* 1, (1948). 122–132.

[10] Alfréd Rényi, *Foundations of Probability*, Holden-Day, 1970.

[11] Neil J. A. Sloane, "Covering arrays and intersecting codes," *J. Combinatorial Designs* 1 (1993) 51–63.

[12] E. Sperner, "Ein Satz über Untermengen einer endlichen Menge," *Math. Z.* 27 (1928) 544–548.

[13] Alan W. Williams, "Determination of test configurations for pair-wise interaction coverage," *Proc. 13th International Conf. on Testing of Communicating Systems*, August, 2000.