Quantum codes do not fix qubit independent errors. Monitoring of calculations in Maxima

J. Lacalle(1), L.M. Pozo-Coronado(2), A.L. Fonseca de Oliveira(3), R. Martín-Cuevas(4)

(1) Universidad Politécnica de Madrid, email: jesus.glopezdelacalle@upm.es

(2) Universidad Politécnica de Madrid, email: lm.pozo@upm.es

(3) Universidad ORT, Montevideo, Uruguay, email: fonseca@ort.edu.uy

(4) Universidad Politécnica de Madrid, email: r.martin-cuevas@alumnos.upm.es

January 5, 2021

Execution time of all instructions: 9:45 minutes (Intel(R) Core(TM) i7-7500U 16GB)

1 Scalar product with complex numbers

```
(%i1) SP(u,v) := conjugate(u).v$
```

2 Error operator of a qubit

```
(%i2) W_1 : matrix([cos(θ0)+%i·sin(θ0)·cos(θ1),-sin(θ0)·sin(θ1)·cos(θ2)+%i·sin(θ0)·sin(θ1)·sin(θ2)],
      [sin(θ0)·sin(θ1)·cos(θ2)+%i·sin(θ0)·sin(θ1)·sin(θ2),cos(θ0)-%i·sin(θ0)·cos(θ1)]);
```

(%o2)

$$\begin{pmatrix} \%i \sin(\theta 0)\cos(\theta 1)+\cos(\theta 0) & \%i \sin(\theta 0)\sin(\theta 1)\sin(\theta 2)-\sin(\theta 0)\sin(\theta 1)\cos(\theta 2) \\ \%i \sin(\theta 0)\sin(\theta 1)\sin(\theta 2)+\sin(\theta 0)\sin(\theta 1)\cos(\theta 2) & \cos(\theta 0)-\%i \sin(\theta 0)\cos(\theta 1) \end{pmatrix}$$

3 Identity operator for a qubit

```
(%i3) I2 : matrix([1,0],[0,1]);
```

(%o3) $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

4 Formula (6) check

```
(%i4) u : [cos(α0)+%i·sin(α0)·cos(α1),sin(α0)·sin(α1)·cos(α2)+%i·sin(α0)·sin(α1)·sin(α2)];
```

(%o4) $[\%i \sin(\alpha 0)\cos(\alpha 1)+\cos(\alpha 0), \%i \sin(\alpha 0)\sin(\alpha 1)\sin(\alpha 2)+\sin(\alpha 0)\sin(\alpha 1)\cos(\alpha 2)]$

```
(%i5) v : W_1.u-u$
```

```
(%i6) Expr_6_1 : expand(SP(v,v))$
```

```
(%i7) Expr_6_2 : expand(subst ([sin(α0)^2=1-cos(α0)^2, sin(α1)^2=1-cos(α1)^2, sin(α2)^2=1-cos(α2)^2,
                 sin(θ0)^2=1-cos(θ0)^2, sin(θ1)^2=1-cos(θ1)^2, sin(θ2)^2=1-cos(θ2)^2], Expr_6_1));
```

(%o7) $2-2\cos(\theta 0)$

5 Formula (9) check

```
(%i8) Expr_9_1 : expand(SP((W_1-I2).[1,0],(W_1-I2).[1,0]));
```

(%o8) $\sin(\theta 0)^2 \sin(\theta 1)^2 \sin(\theta 2)^2+\sin(\theta 0)^2 \sin(\theta 1)^2 \cos(\theta 2)^2+\sin(\theta 0)^2 \cos(\theta 1)^2+\cos(\theta 0)^2-2\cos(\theta 0)+1$

```
(%i9) Expr_9_2 : expand(subst ([sin(θ0)^2=1-cos(θ0)^2, sin(θ1)^2=1-cos(θ1)^2, sin(θ2)^2=1-cos(θ2)^2],
        Expr_9_1));
```

(%o9) $2-2\cos(\theta 0)$

```
(%i10) expand(SP((W_1-I2).[1,0],(W_1-I2).[0,1]));
```

(%o10) 0

```
(%i11) expand(SP((W_1-I2).[0,1],(W_1-I2).[1,0]));
```

(%o11) 0

```
(%i12) Expr_9_3 : expand(SP((W_1-I2).[0,1],(W_1-I2).[0,1]));
```

(%o12) $\sin(\theta 0)^2 \sin(\theta 1)^2 \sin(\theta 2)^2+\sin(\theta 0)^2 \sin(\theta 1)^2 \cos(\theta 2)^2+\sin(\theta 0)^2 \cos(\theta 1)^2+\cos(\theta 0)^2-2\cos(\theta 0)+1$

```
(%i13) Expr_9_4 : expand(subst ([sin(θ0)^2=1-cos(θ0)^2, sin(θ1)^2=1-cos(θ1)^2, sin(θ2)^2=1-cos(θ2)^2],
        Expr_9_3));
```

(%o13) $2-2\cos(\theta 0)$

6 Rest of Pauli operators (matrices)

```
(%i14) X : matrix([0,1],[1,0]);
```

(%o14) $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

```
(%i15) Y : matrix([0,-%i],[%i,0]);
```

$$(\%o15)\quad \begin{pmatrix} 0 & -\%i \\ \%i & 0 \end{pmatrix}$$

```
(%i16) Z : matrix([1,0],[0,-1]);
```

$$(\%o16)\quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

7 Tensor product

```
(%i17) TP(M1,M2) := block([Size, r1, c1, r2, c2, r, c, M, j], Size:matrix_size(M1), r1:Size[1], c1:Size[2],
           Size:matrix_size(M2), r2:Size[1], c2:Size[2], r:r1·r2, c:c1·c2, M:zeromatrix(r,c),
        for j from 1 thru r do block([k], for k from 1 thru c do
             M[j,k]:M1[1 + floor((j - 1)/r2), 1 + floor((k - 1)/c2)]·
             M2[1 + mod(j - 1, r2), 1 + mod(k - 1, c2)]),M)$
```

8 Discrete errors fixed by 5-qubit quantum code

```
(%i18) I : TP(TP(TP(TP(I2,I2),I2),I2),I2)$

(%i19) X0 : TP(TP(TP(TP(X,I2),I2),I2),I2)$

(%i20) X1 : TP(TP(TP(TP(I2,X),I2),I2),I2)$

(%i21) X2 : TP(TP(TP(TP(I2,I2),X),I2),I2)$

(%i22) X3 : TP(TP(TP(TP(I2,I2),I2),X),I2)$

(%i23) X4 : TP(TP(TP(TP(I2,I2),I2),I2),X)$
```

```
(%i24) Y0 : TP(TP(TP(TP(Y,I2),I2),I2),I2)$

(%i25) Y1 : TP(TP(TP(TP(I2,Y),I2),I2),I2)$

(%i26) Y2 : TP(TP(TP(TP(I2,I2),Y),I2),I2)$

(%i27) Y3 : TP(TP(TP(TP(I2,I2),I2),Y),I2)$

(%i28) Y4 : TP(TP(TP(TP(I2,I2),I2),I2),Y)$

(%i29) Z0 : TP(TP(TP(TP(Z,I2),I2),I2),I2)$

(%i30) Z1 : TP(TP(TP(TP(I2,Z),I2),I2),I2)$

(%i31) Z2 : TP(TP(TP(TP(I2,I2),Z),I2),I2)$

(%i32) Z3 : TP(TP(TP(TP(I2,I2),I2),Z),I2)$

(%i33) Z4 : TP(TP(TP(TP(I2,I2),I2),I2),Z)$

(%i34) Discrete_Error : [I,X0,X1,X2,X3,X4,Y0,Y1,Y2,Y3,Y4,Z0,Z1,Z2,Z3,Z4]$
```

9 5-qubit quantum code generators

```
(%i35) v0 : (1/4)·[1,0,0,-1,0,1,-1,0,0,1,1,0,-1,0,0,-1,0,-1,1,0,1,0,0,-1,-1,0,0,-1,0,-1,-1,0]$

(%i36) v1 : (1/4)·[0,-1,-1,0,-1,0,0,-1,-1,0,0,1,0,1,-1,0,-1,0,0,-1,0,1,1,0,0,-1,1,0,-1,0,0,1]$
```

10 Basis change matrix from B (basis associated with the code) to B_C

```
(%i37) M_aux : zeromatrix(32,32)$

(%i38) for k from 1 thru 16 do block (M_aux[2·k−1] : flatten(args(Discrete_Error[k].v0)),
           M_aux[2·k] : flatten(args(Discrete_Error[k].v1)))$
```

M is the unitary basis change matrix

```
(%i39) M : transpose(M_aux)$
```

We verify that indeed M is unitary

```
(%i40) M_aux : M.transpose(conjugate(M))$

(%i41) if M_aux = diagmatrix(32,1) then "unitary" else "not unitary";
(%o41)  unitary

(%i42) M_aux : transpose(conjugate(M)).M$

(%i43) if M_aux = diagmatrix(32,1) then "unitary" else "not unitary";
(%o43)  unitary
```

11 Error operator on each qubit

```
(%i44) W0 : matrix([A0+%i·B0,−C0+%i·D0],[C0+%i·D0,A0−%i·B0])$

(%i45) W1 : matrix([A1+%i·B1,−C1+%i·D1],[C1+%i·D1,A1−%i·B1])$
```

```
(%i46) W2 : matrix([A2+%i·B2,−C2+%i·D2],[C2+%i·D2,A2−%i·B2])$
```

```
(%i47) W3 : matrix([A3+%i·B3,−C3+%i·D3],[C3+%i·D3,A3−%i·B3])$
```

```
(%i48) W4 : matrix([A4+%i·B4,−C4+%i·D4],[C4+%i·D4,A4−%i·B4])$
```

12 Error operator in the 5 qubits

```
(%i49) W : TP(TP(TP(TP(W0,W1),W2),W3),W4)$
```

Error operator in the 5 qubits with respect to the code basis

```
(%i50) W_B : transpose(conjugate(M)).W.M$
```

13 Coordinates of the Phi_B and Psi_B vectors

```
(%i51) φ(w0,w1,w2,w3) := [w0+%i·w1,w2+%i·w3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$
```

```
(%i52) Ψ(w0,w1,w2,w3) := W_B.φ(w0,w1,w2,w3)$
```

14 Lemma 1 check

```
(%i53) a(s) := realpart(Ψ(1,0,0,0)[2·s+1][1])$
```

```
(%i54) b(s) := imagpart(Ψ(1,0,0,0)[2·s+1][1])$
```

```
(%i55) c(s) := realpart(Ψ(0,0,1,0)[2·s+1][1])$
```

```
(%i56) d(s) := imagpart(Ψ(0,0,1,0)[2·s+1][1])$
```

For even s (16 equalities and Ψ(w0,w1,w2,w3) coordinates start at 1)

```
(%i57) Expr_lem1_1(s) := (a(s)+%i·b(s))·w0+(-b(s)+%i·a(s))·w1+(c(s)+%i·d(s))·w2+(-d(s)+%i·c(s))·w3$

(%i58) sum(if expand(Ψ(w0,w1,w2,w3)[2·s+1][1] - Expr_lem1_1(s)) = 0 then 1 else 0,s,0,15);
(%o58) 16
```

For s = 1 (1 equality and Ψ(w0,w1,w2,w3) coordinates start at 1)

```
(%i59) Expr_lem1_2 : (-c(0)+%i·d(0))·w0+(-d(0)-%i·c(0))·w1+(a(0)-%i·b(0))·w2+(b(0)+%i·a(0))·w3$

(%i60) if expand(Ψ(w0,w1,w2,w3)[2][1] - Expr_lem1_2) = 0 then 1 else 0;
(%o60) 1
```

For odd s (15 equalities and Ψ(w0,w1,w2,w3) coordinates start at 1)

```
(%i61) Expr_lem1_3(s) := (c(s)-%i·d(s))·w0+(d(s)+%i·c(s))·w1+(-a(s)+%i·b(s))·w2+(-b(s)-%i·a(s))·w3$

(%i62) sum(if expand(Ψ(w0,w1,w2,w3)[2·s+2][1] - Expr_lem1_3(s)) = 0 then 1 else 0,s,1,15);
(%o62) 15
```

15 Proof of Lemma 2 Check

```
(%i63) Expr_lem2_0 : (a+%i·b)·w0+(-b+%i·a)·w1+(c+%i·d)·w2+(-d+%i·c)·w3$

(%i64) Expr_lem2_1 : (c-%i·d)·w0+(d+%i·c)·w1+(-a+%i·b)·w2+(-b-%i·a)·w3$

(%i65) expand(Expr_lem2_0·conjugate(Expr_lem2_0)+Expr_lem2_1·conjugate(Expr_lem2_1)-
          (a^2+b^2+c^2+d^2)·(w0^2+w1^2+w2^2+w3^2));
(%o65) 0
```

16 Functions to analyze the monomials associated with the coordinates of Ψ

var_list is the list of variables

```
(%i66) var_list:[A0,A1,A2,A3,A4,B0,B1,B2,B3,B4,C0,C1,C2,C3,C4,D0,D1,D2,D3,D4]$
```

var_set is the set of variables

```
(%i67) var_set : {A0,A1,A2,A3,A4,B0,B1,B2,B3,B4,C0,C1,C2,C3,C4,D0,D1,D2,D3,D4}$
```

var_list_index is the list of sets of variables with subscript 0, 1, 2, 3 and 4 respectively

```
(%i68) var_set_index : [{A0,B0,C0,D0},{A1,B1,C1,D1},{A2,B2,C2,D2},{A3,B3,C3,D3},{A4,B4,C4,D4}]$
```

var_set_A is the set variables with letter A

```
(%i69) var_set_A : {A0,A1,A2,A3,A4}$
```

var(m,v) returns {v} if the variable v is in the monomial m and {} otherwise

```
(%i70) var(m,v) :=
          if not coeff(m,v) = 0
             then {v}
             else {}$
```

set_of_var(m) returns the set of variables of the monomial m

```
(%i71) set_of_var(m) := block([r:{}, x],
         for x in var_list do r : union(var(m,x),r),
         r)$
```

list_of_mon(p) calculates a list with the sets of variables of the monomials of the polynomial p

```
(%i72) list_of_mon(p) := block([r:[], x, list_mon], list_mon : args(p),
         for x in list_mon do r : flatten([r,set_of_var(x)]),
         r)$
```

set_of_mon(p) calculates a set whose elements are the sets of variables of the monomials of the polynomial p

```
(%i73) set_of_mon(p) := block([r:{}, x, list_mon], list_mon : args(p),
         for x in list_mon do r : union(r,{set_of_var(x)}),
         r)$
```

17 Lists of variable sets of a(s), b(s), c(s), and d(s)

```
(%i74) list_of_mon_a() := block([r, s], r : [list_of_mon(expand(a(0)))],
           for s from 1 thru 15 do r : append(r,[list_of_mon(expand(a(s)))]),
         r)$
```

```
(%i75) List_a : list_of_mon_a()$
```

```
(%i76) list_of_mon_b() := block([r, s], r : [list_of_mon(expand(b(0)))],
           for s from 1 thru 15 do r : append(r,[list_of_mon(expand(b(s)))]),
         r)$
```

```
(%i77) List_b : list_of_mon_b()$

(%i78) list_of_mon_c() := block([r, s], r : [list_of_mon(expand(c(0)))],
          for s from 1 thru 15 do r : append(r,[list_of_mon(expand(c(s)))]),
        r)$

(%i79) List_c : list_of_mon_c()$

(%i80) list_of_mon_d() := block([r, s], r : [list_of_mon(expand(d(0)))],
          for s from 1 thru 15 do r : append(r,[list_of_mon(expand(d(s)))]),
        r)$

(%i81) List_d : list_of_mon_d()$
```

18 Checking Lemma 3, item 1

a(s) is a homogeneous polynomial of degree 5 in the set of variables
var_set (16*16 equalities)

```
(%i82) sum(sum(if cardinality(intersection(List_a[s+1][k],var_set)) = 5 then 1 else 0,k,1,16),s,0,15);
(%o82) 256
```

b(s) is a homogeneous polynomial of degree 5 in the set of variables
var_set (16*16 equalities)

```
(%i83) sum(sum(if cardinality(intersection(List_b[s+1][k],var_set)) = 5 then 1 else 0,k,1,16),s,0,15);
(%o83) 256
```

c(s) is a homogeneous polynomial of degree 5 in the set of variables
var_set (16*16 equalities)

```
(%i84) sum(sum(if cardinality(intersection(List_c[s+1][k],var_set)) = 5 then 1 else 0,k,1,16),s,0,15);
(%o84) 256
```

d(s) is a homogeneous polynomial of degree 5 in the set of variables
var_set (16*16 equalities)

```
(%i85) sum(sum(if cardinality(intersection(List_d[s+1][k],var_set)) = 5 then 1 else 0,k,1,16),s,0,15);
(%o85) 256
```

a(s) is a homogeneous polynomial of degree 1 in the set of variables
with subscript 0, 1, 2, 3 and 4 (16*16*5 equalities)

```
(%i86) sum(sum(sum(if cardinality(intersection(List_a[s+1][k],var_set_index[j])) = 1 then 1 else 0,k,1,16)
        ,s,0,15),j,1,5);
(%o86) 1280
```

b(s) is a homogeneous polynomial of degree 1 in the set of variables
with subscript 0, 1, 2, 3 and 4
(16*16*5 equalities)

```
(%i87) sum(sum(sum(if cardinality(intersection(List_b[s+1][k],var_set_index[j])) = 1 then 1 else 0,k,1,16)
        ,s,0,15),j,1,5);
(%o87) 1280
```

c(s) is a homogeneous polynomial of degree 1 in the set of variables
with subscript 0, 1, 2, 3 and 4
(16*16*5 equalities)

```
(%i88) sum(sum(sum(if cardinality(intersection(List_c[s+1][k],var_set_index[j])) = 1 then 1 else 0,k,1,16)
        ,s,0,15),j,1,5);
(%o88) 1280
```

d(s) is a homogeneous polynomial of degree 1 in the set of variables
with subscript 0, 1, 2, 3 and 4
(16*16*5 equalities)

```
(%i89) sum(sum(sum(if cardinality(intersection(List_d[s+1][k],var_set_index[j])) = 1 then 1 else 0,k,1,16)
        ,s,0,15),j,1,5);
(%o89) 1280
```

Each of the expressions a(s) have 16 monomials (16 equalities)

```
(%i90) sum(if length(List_a[s+1]) = 16 then 1 else 0,s,0,15);
(%o90) 16
```

Each of the expressions b(s) have 16 monomials (16 equalities)

```
(%i91) sum(if length(List_b[s+1]) = 16 then 1 else 0,s,0,15);
(%o91) 16
```

Each of the expressions c(s) have 16 monomials (16 equalities)

```
(%i92) sum(if length(List_c[s+1]) = 16 then 1 else 0,s,0,15);
(%o92) 16
```

Each of the expressions d(s) have 16 monomials (16 equalities)

```
(%i93) sum(if length(List_d[s+1]) = 16 then 1 else 0,s,0,15);
(%o93) 16
```

19 Checking Lemma 3, item 2

Set of monomials of a(s) for all 0 ≤ s < 16

```
(%i94) set_of_mon_a() := block([r, s], r : set_of_mon(expand(a(0))),
           for s from 1 thru 15 do r : union(r,set_of_mon(expand(a(s)))),
         r)$
```

```
(%i95) Set_a : set_of_mon_a()$
```

Set of monomials of b(s) for all 0 ≤ s < 16

```
(%i96) set_of_mon_b() := block([r, s], r : set_of_mon(expand(b(0))),
           for s from 1 thru 15 do r : union(r,set_of_mon(expand(b(s)))),
         r)$
```

```
(%i97) Set_b : set_of_mon_b()$
```

Set of monomials of c(s) for all 0 ≤ s < 16

```
(%i98) set_of_mon_c() := block([r, s], r : set_of_mon(expand(c(0))),
           for s from 1 thru 15 do r : union(r,set_of_mon(expand(c(s)))),
         r)$
```

```
(%i99) Set_c : set_of_mon_c()$
```

Set of monomials of d(s) for all 0 ≤ s < 16

```
(%i100) set_of_mon_d() := block([r, s], r : set_of_mon(expand(d(0))),
           for s from 1 thru 15 do r : union(r,set_of_mon(expand(d(s)))),
         r)$
```

```
(%i101) Set_d : set_of_mon_d()$
```

Set of all monomials

```
(%i102) Set_of_all_mon : union(union(union(Set_a,Set_b),Set_c),Set_d)$
```

All monomials are different because the cardinal of the union is the sum of the cardinals

```
(%i103) length(Set_of_all_mon);
(%o103) 1024
```

20 Checking Lemma 3, item 3

Check for monomials of a(s) (2*(16 choose 2)*16 = 3840 equalities)

```
(%i104) sum(sum(sum(if cardinality(intersection(List_a[s+1][j],List_a[s+1][k])) = 1 then 1 else 0,j,1,16),
        k,1,16),s,0,15);
(%o104) 3840
```

Check for monomials of b(s) (2*(16 choose 2)*16 = 3840 equalities)

```
(%i105) sum(sum(sum(if cardinality(intersection(List_b[s+1][j],List_b[s+1][k])) = 1 then 1 else 0,j,1,16),
        k,1,16),s,0,15);
(%o105) 3840
```

Check for monomials of c(s) (2*(16 choose 2)*16 = 3840 equalities)

```
(%i106) sum(sum(sum(if cardinality(intersection(List_c[s+1][j],List_c[s+1][k])) = 1 then 1 else 0,j,1,16),
        k,1,16),s,0,15);
(%o106) 3840
```

Check for monomials of d(s) (2*(16 choose 2)*16 = 3840 equalities)

```
(%i107) sum(sum(sum(if cardinality(intersection(List_d[s+1][j],List_d[s+1][k])) = 1 then 1 else 0,j,1,16),
        k,1,16),s,0,15);
(%o107) 3840
```

21 Table (13) check

Set_var(s) is the set of variables of P_s, that is, the variables of
a(s), b(s), c(s) and d(s)

```
(%i108) Set_var(s) := union(union(union(set_of_mon(expand(a(s))),set_of_mon(expand(b(s))),
            set_of_mon(expand(c(s))),set_of_mon(expand(d(s))))))$
```

class_A_var(Set_mon) calculates the number of monomials in Set_mon
with 0 A's, 1 A, 2 A's, 3 A's, 4 A's and 5 A's

```
(%i109) class_A_var(Set_mon) := block([r, x, y], r:[0,0,0,0,0,0],
    for x in Set_mon do block(y : cardinality(intersection(x,var_set_A)), r[y+1] : r[y+1]+1),
  r)$
```

Table (13) for P_0 (1 equality)

```
(%i110) if class_A_var(Set_var(0)) = [18,15,30,0,0,1] then 1 else 0;
(%o110) 1
```

Table (13) for P_s where 0 < s < 16 (15 equalities)

```
(%i111) sum(if class_A_var(Set_var(s)) = [15,26,16,6,1,0] then 1 else 0,s,1,15);
(%o111) 15
```

22 Sets V_0 to V_15 check

Sets that satisfy the required properties

```
(%i112) Set_V : [{A0, B0, B3, B4, C2, D2, D3, D4},
          {B3, B4, C0, C2, D0, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B3, B4, C0, C2, D0, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B3, B4, C0, C2, D0, D2, D3, D4},
          {B0, B4, C1, C3, D0, D1, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B0, B4, C1, C3, D0, D1, D3, D4},
          {B0, B1, C2, C4, D0, D1, D2, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B2, B3, C1, C4, D1, D2, D3, D4},
          {B3, B4, C0, C2, D0, D2, D3, D4}]$
```

Verification of the intersection of each set of Set_V[s+1] with all monomials of a(s), b(s), c(s) and d(s) (4*16*16 = 1024 equalities)

```
(%i113) sum(sum(mod(cardinality(intersection(Set_V[s+1],List_a[s+1][k])),2)+
              mod(cardinality(intersection(Set_V[s+1],List_b[s+1][k])),2)+
              mod(cardinality(intersection(Set_V[s+1],List_c[s+1][k])),2)+
              mod(cardinality(intersection(Set_V[s+1],List_d[s+1][k])),2),s,0,15),k,1,16);
(%o113) 1024
```

23 Sets V_0' to V_4' check

Sets that satisfy the required properties

```
(%i114) Set_V_prime : [{A0, B0, B3, B4, C2, D2, D3, D4},
        {A1, B1, B2, B3, C4, D2, D3, D4},
        {A2, B0, C0, C3, C4, D2, D3, D4},
        {A3, B1, B2, B4, C2, D1, D3, D4},
        {A4, B1, C1, C2, C3, D2, D3, D4}]$
```

Verification of the intersection of each set of Set_V_prime[s+1] with all monomials of a(0), b(0), c(0) and d(0) (5*4*16 = 320 equalities)

```
(%i115) sum(sum(mod(cardinality(intersection(Set_V_prime[s+1],List_a[1][k])),2)+
          mod(cardinality(intersection(Set_V_prime[s+1],List_b[1][k])),2)+
          mod(cardinality(intersection(Set_V_prime[s+1],List_c[1][k])),2)+
          mod(cardinality(intersection(Set_V_prime[s+1],List_d[1][k])),2),s,0,4),k,1,16);
(%o115) 320
```

Verification of the intersection of each set of Set_V_prime[s+1] with the set of variables of the monomial A0*A1*A2*A3*A4

```
(%i116) intersection(Set_V_prime[1],var_set_A);
(%o116) {A0}

(%i117) intersection(Set_V_prime[2],var_set_A);
(%o117) {A1}

(%i118) intersection(Set_V_prime[3],var_set_A);
(%o118) {A2}
```

```
(%i119) intersection(Set_V_prime[4],var_set_A);
(%o119) {A3}

(%i120) intersection(Set_V_prime[5],var_set_A);
(%o120) {A4}
```

24 Sets V_1'' to V_15'' check

Sets that satisfy the required properties, preceded by the sets of variables of the corresponding monomials of a0

```
(%i121) Set_V_prime_prime : [[{A0,B1,B4,C2,C3},{A1, B1, B2, B3, C4, D2, D3, D4}],
        [{A0,B2,B3,D1,D4},{A1, B2, C0, C1, C3, D0, D2, D3}],
        [{A0,C1,C4,D2,D3},{A1, B2, B3, B4, C1, C2, C3, D4}],
        [{A1,B0,B2,C3,C4},{A0, B0, B3, B4, C2, D2, D3, D4}],
        [{A1,B3,B4,D0,D2},{A0, B0, B1, B3, C1, C3, C4, D4}],
        [{A1,C0,C2,D3,D4},{A0, B2, B3, B4, C0, C3, C4, D2}],
        [{A2,B0,B4,D1,D3},{A0, B0, B1, B3, C1, C3, C4, D4}],
        [{A2,B1,B3,C0,C4},{A0, B1, B2, B4, C1, D0, D2, D4}],
        [{A2,B1,B3,C0,C4},{A0, B1, B2, B4, C1, D0, D2, D4}],
        [{A3,B0,B1,D2,D4},{A0, B0, B2, B4, C1, C2, C4, D1}],
        [{A3,B2,B4,C0,C1},{A0, B2, C2, C3, C4, D0, D3, D4}],
        [{A3,C2,C4,D0,D1},{A0, B0, B3, B4, C2, D2, D3, D4}],
        [{A4,B0,B3,C1,C2},{A0, B0, B1, B2, C3, D1, D2, D3}],
        [{A4,B1,B2,D0,D3},{A0, B1, C0, C2, C4, D1, D2, D4}],
        [{A4,C0,C3,D1,D2},{A0, B1, B2, B3, C0, C1, C2, D3}]]$
```

Verification of the intersection of each set of Set_V_prime_prime[s][2] with all monomials of a(0), b(0), c(0) and d(0) (15*4*16 = 960 equalities)

```
(%i122) sum(sum(mod(cardinality(intersection(Set_V_prime_prime[s][2],List_a[1][k])),2)+
        mod(cardinality(intersection(Set_V_prime_prime[s][2],List_b[1][k])),2)+
        mod(cardinality(intersection(Set_V_prime_prime[s][2],List_c[1][k])),2)+
        mod(cardinality(intersection(Set_V_prime_prime[s][2],List_d[1][k])),2),s,1,15),k,1,16);
(%o122) 960
```

Verification of the intersection of each set of Set_V_prime_prime[s][2] with the set of variables of the corresponding monomial of a(0)

```
(%i123) Intersect_V_prime_prime_a0() := block([r, s],
        r : [intersection(Set_V_prime_prime[1][2],Set_V_prime_prime[1][1])],
        for s from 2 thru 15 do
        r : append(r,[intersection(Set_V_prime_prime[s][2],Set_V_prime_prime[s][1])]),
        r)$
```

```
(%i124) Intersect_V_prime_prime_a0();
(%o124) [{B1},{B2},{C1},{B0},{B3},{C0},{B0},{B1},{B1},{B0},{B2},{C2},{B0},{B1},{
C0}]
```

25 End of document