Beiträge zur Algebra und Geometrie
14 (1983), 15–21

Lattice Generation Program for Computing a Vector Space Lattice

LENNI HAAPASALO and PEKKA NIEMISTÖ

Introduction

We consider a sesquilinear space[1]) (E, Φ) of countable dimension; the form Φ is assumed to be nondegenerate and orthosymmetric. By the orthostable lattice $\vartheta(F_1, \ldots, F_n)$ generated by the given subspaces $F_1, \ldots, F_n$ of E we mean the smallest sublattice of $\mathscr{L}(E)$ containing $F_1, \ldots, F_n$, (0) and E and such that $X^\perp \in \vartheta$ whenever $X \in \vartheta$. The importance of these orthostable lattices, which have taken a central role in the theorems of Witt type, can easily been illustrated by the following example: Given two families $(F_1, \ldots, F_n)$ and $(\bar{F}_1, \ldots, \bar{F}_n)$ of subspaces of E we can ask if there exists an isometry $T: E \to E$ such that $TF_i = \bar{F}_i$ for each i. Such an isometry — if it exists — induces a lattice isomorphism $\tau : \vartheta(F_1, \ldots, F_n) \to \bar{\vartheta}(\bar{F}_1, \ldots, \bar{F}_n)$ which commutes with orthogonal complementation. Furthermore, this ortho-isomorphismus preserves indices, i.e. dimensions of quotient spaces X/Y of neighbouring elements $X, Y \in \vartheta$. If the space is alternate, these indices form a complete set of orthogonal invariants for a subspace F of E. This can be proved by applying the remarkable theorem of GROSS (theorem IV.1 in [2]) to the well known lattice of KAPLANSKY with fourteen elements. The same theorem gives us, in some special cases, an immediate answer to the question asked above; namely whenever the lattice $\vartheta(F_1, \ldots, F_n)$ happens to be finite and distributive, there exists an isometry $T: E \to E$ which has the property $TF_i = \bar{F}_i$ for all i. Generally, however, the lattice ϑ is infinite and nondistributive — such is the case even with the lattice generated by two subspaces. Now there arise questions of following kind: How does one know when the lattice attached to the problem will be finite and what can be said about its diagram? These questions have no easy answer. For some special classes of subspaces it has been possible to solve the problem of orthogonal classification without computing the lattice explicitely (for example in case the two subspaces can be separated orthogonally in E). In most cases, especially in non-distributive cases, a thorough knowledge of the lattice is necessary. One then has to make a plan for computing the lattice that belongs to the problem as "economically" as possible. By this economizing we mean a method which takes at each step of the calculation information from the indermediate states as much as possible. This principle is particularly useful if the lattice has some finite indices, because the orthostability can then in several cases be omitted (see [3]).
This paper is concerned with an effort to compute the orthostable lattice $\vartheta(F_1, \ldots, F_n)$ with a computer. The program will give a possibility to switch off orthogonalization,

[1]) We remark that the terminology used here is found in [2].

if orthostability is supposed to be omitted. We wish to remark that no attempts have been made to generalize the method for non trace-valued forms even when there was such a possibility.
The authors are indebted to H. GROSS for suggesting the study of nondistributive lattices which has inspirated our computational methods.

1. A survey of the program structure

In this section the principle of the used PL/1 lattice generation program is briefly described. We do not pay here any attention to a detailed explanation of the resolution rules, say programming of set theoretic deductions etc.
As an input to the program any subspaces $F_1, \ldots, F_n$ of E with the necessary assumptions can be given. We call them *basic spaces* because of the fact that the given information on them is carried immediately on to the data area, which we call here throughout *data base*. The spaces are represented all the time during the process by consecutive integers as follows: $1 := (0)$, $2 := E$, $3 := F_1, \ldots, n + 2 := F_n, \ldots$ etc. By input it is possible, of course, to give storeable information about the spaces $n + 3$, $n + 4, \ldots$ to be generated, as well as to switch off the operation $\perp$ or to fix the maximal number of the spaces to be computed.
As an output the program lists the operation matrix (i.e. a matrix with the sum spaces in the upper triangle and intersection spaces in the lower one) including a column of the orthogonal complements, the inclusion matrix and, finally, the immediate successors among the lattice elements.

1.1. The main program

The method of the Lattice Generation Program is based on the fact that the elements of a finite vector space lattice can be written as a sequence $(0), E, F_1, F_2, \ldots, F_k, F_{k+1}, \ldots, F_n$ so that $F_1, \ldots, F_k$ are the basic spaces and for every $k < p < n$ the subspace F_p has some of the following representations

$$F_p = F_i^{\perp}, \text{ for some } i < p,$$
$$F_p = F_i + F_j, \text{ for some } i, j < p,$$
$$F_p = F_i \cap F_j, \text{ for some } i, j < p.$$

The program generates a sequence of this kind. During the execution the program knows a part of this sequence, say $(0), E, F_1, \ldots, F_i$. Now an element *current* is generated, and if it turns out to be a new element, it is inserted into the sequence as an element F_{i+1}. The program performs the following steps:

```
A 1. Create the trivial elements (0) and $E$.
A 2. Read the information on the basic spaces and create these elements.
A 3. If all possible generations are performed, the lattice is ready and the program
     stops.
A 4. Generate a space using the existing elements and create a temporary element
     current.
A 5. Search, if current exists in data base:
     A 5.1. If current is found to exist as an element $A$, then move current in-
            formation to $A$ and delete current.
     A 5.2. Else set current into the data base as a new element.
A 6. Continue from step A 3.
```

1.2. The data base and the data base programs

All information of the spaces is collected in the data base, which contains the *operation matrix* OPERMAT, the *inclusion matrix* INCLUS and the following three vectors: ORTOG, CLOSE and DIMENS. These data areas deliver information as follows:

$$\mathrm{OPERMAT}(I, J) = \begin{cases} \text{the number of the sum space } I + J, \text{ if } I < J \\ \text{the number of the intersection space } I \cap J, \text{ if } I > J \\ 0, \text{ if } I = J \end{cases}$$

$\mathrm{INCLUS}(I, J) = 1$, if $I \subset J$, else 0,

$\mathrm{ORTOC}(I) = I^{\perp}$,

$\mathrm{CLOSE}(I) = 1$, if I is orthogonally closed, else 0,

$\mathrm{DIMENS}(I) = 1$, if I is finite dimensional, else 0.

These data areas are filled during the execution and they are mainly used by the special subroutines. These recursive subroutines are called *data base programs*, they search and update the base. The fundamental set theory logic is included in these programs. Whenever some information is inserted into the data base, the subroutine takes care that all the implications of this information are also carried into the data base. For example, if $A \subset B$ is noticed, then the subroutine calls another program, that sets $A = A + B$; furthermore a later program may call some others etc. Hence, a small information can make large changes in the data base.

2. Some applications and comments

Let us light up possibilities for applying the program by the following three examples.

Example 1. If F and G are two subspaces of E with the assumptions

$$F^{\perp} = G^{\perp} = (0), \tag{1}$$

$$(F \cap G)^{\perp} \subset F \cap G, \tag{2}$$

and

$$\left(F + (F \cap G)^{\perp\perp}\right) \cap \left(G + (F \cap G)^{\perp\perp}\right) \tag{3}$$

is closed, we can ask if there exists an isometry $T: E \to E$ such that $TF = F$ and $TG = G$. In order to solve this problem we first have to compute the lattice $\vartheta(F, G)$. The operation matrix of this is shown in table 1 and the lattice has the form given in figure 2. It turns out that our problem is solvable if we assume that dim (14/22) $< \infty$. This requirement could be observed only after exactly knowing $\vartheta(F, G)$ so in this case it is not possible to economize in the computing process.

Example 2. Let be given the two chains $F_1 \subset F_2 \subset F_3$ and $F_4 \subset F_5 \subset F_6$ of subspaces of E. These chains generate a distributive lattice which has $(4 + 4)!/4!4! = 70$ elements (see [1] p. 66), if only the operations $+$ and $\cap$ are taken. The lattice takes a form of figure 3.

Example 3. As a third application of the program we wish to point out a situation which is typical of building the lattice attached to the problem. Suppose we already have computed a great lattice ϑ_1 containing an element X which is not orthostable.

If we have to combine $X^{\perp}$ with the elements of ϑ_1 it is often advandageous to first limit the examination to a certain sublattice, instead of trying to compute the whole lattice at one blow. Let us consider the diagram below:

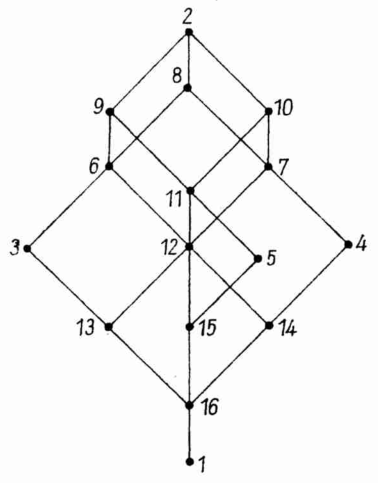


Fig. 1

Table 1

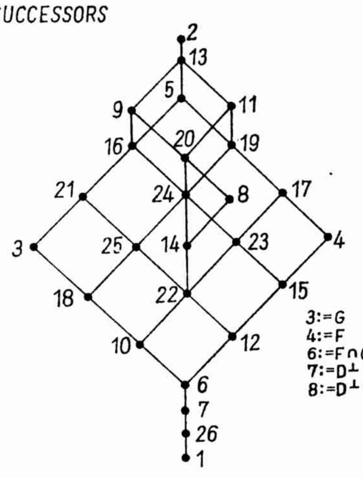
Operation matrix:

X $X^{\perp}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
3	1	0	5	5	3	3	9	9	3	13	21	13	16	16	16	5	3	5	9	21	21	16	16	21	3
4	1	6	0	5	4	4	11	13	17	11	4	13	19	4	5	17	19	19	11	5	17	17	19	19	4
5	1	3	4	0	5	5	13	13	5	13	5	13	5	5	5	5	5	5	13	5	5	5	5	5	5
6	7	6	6	6	0	6	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	6
7	8	7	7	7	7	0	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	7
8	7	10	12	14	6	7	0	9	8	11	8	13	8	20	9	11	20	11	20	9	8	20	20	20	8
9	1	3	15	16	6	7	8	0	9	13	9	13	9	9	9	13	9	13	9	9	9	9	9	9	9
10	7	10	6	10	6	7	10	10	0	11	22	13	14	23	16	17	18	19	20	21	22	23	24	25	10
11	1	18	4	19	6	7	8	20	10	0	11	13	11	11	13	11	11	11	11	13	11	11	11	11	11
12	7	6	12	12	6	7	12	12	6	12	0	13	14	15	16	17	25	19	20	21	22	23	24	25	12
13	1	3	4	5	6	7	8	9	10	11	12	0	13	13	13	13	13	13	13	13	13	13	13	13	13
14	7	10	12	14	6	7	14	14	10	14	12	14	0	24	16	19	24	19	20	16	14	24	24	24	14
15	26	6	15	15	6	7	12	15	6	15	12	15	12	0	16	17	24	19	20	16	23	23	24	24	15
16	1	3	15	16	6	7	14	16	10	24	12	16	14	15	0	5	16	5	9	16	16	16	16	16	16
17	1	10	4	17	6	7	22	23	10	17	12	17	22	15	23	0	19	19	11	5	17	17	19	19	17
18	26	18	6	18	6	7	10	18	10	18	6	18	10	6	18	10	0	19	20	21	25	24	24	25	18
19	1	18	4	19	6	7	14	24	10	19	12	19	14	15	24	17	18	0	11	5	19	19	19	19	19
20	26	18	15	24	6	7	6	20	10	20	12	20	14	15	24	23	18	24	0	9	20	20	20	20	20
21	1	3	12	21	6	7	22	21	10	25	12	21	22	12	21	22	18	25	25	0	21	16	16	21	21
22	7	10	12	22	6	7	22	22	10	22	12	22	22	12	22	22	10	22	22	22	0	23	24	25	22
23	26	10	15	23	6	7	22	23	10	23	12	23	22	15	23	23	10	23	23	22	22	0	24	24	23
24	26	18	15	24	6	7	14	24	10	24	12	24	14	15	24	23	16	24	24	25	22	23	0	24	24
25	26	16	12	25	6	7	22	25	10	25	12	25	22	12	25	22	18	25	25	25	22	22	25	0	25
26	20	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	0

THE IMMEDIATE SUCCESSORS

1 :	26
2 :	
3 :	21
2 :	17
5 :	13
6 :	10 12
7 :	6
8 :	20
9 :	13
10 :	18 22
11 :	13
12 :	15 22
13 :	2
14 :	8 24
15 :	4 23
16 :	5 9
17 :	19
18 :	3 25
19 :	5 11
20 :	9 11
21 :	16
22 :	14 23 25
23 :	17 24
24 :	16 19 20
25 :	21 24
26 :	7



3:=G
4:=F
6:=F∩G=:D
7:=D$^{\perp}$
8:=D$^{\perp\perp}$

Fig. 2

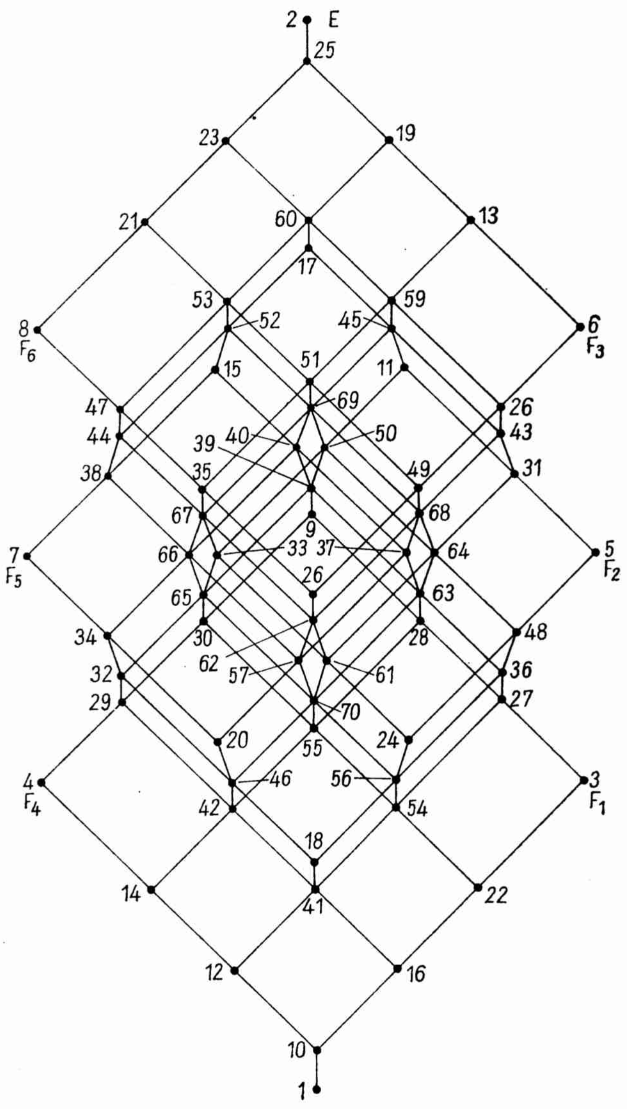


Fig. 3

If we want to join an element 17 such that $5 \subset 17$ and $3 + 17 = 4 + 17 = 2$, it is difficult to imagine at once the form that the resulting lattice is going to have. Figure 4 shows, that the result is quite complicated.

At first our program was planned for computing orthostable lattices only. However, after the innovation that orthostability could be omitted, applications of the type of examples 2 and 3 arise very often. It would therefore be reasonable to change the program in many ways. For Univac 1100 computer it takes only some minutes to compute a lattice about 30—40 elements, but the execution time grows very fast for greater lattices.

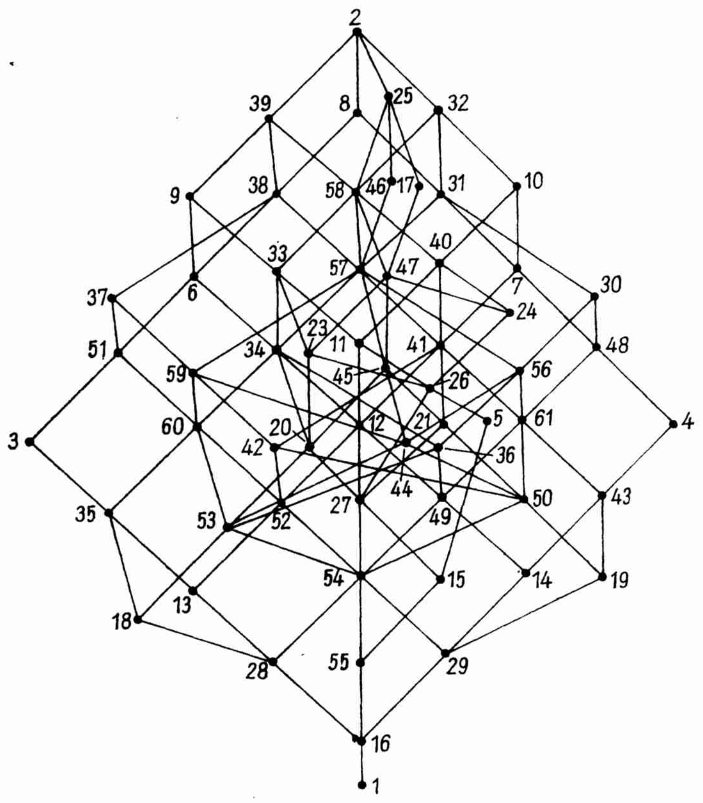


Fig. 4

The reliability of the program can be proved only "in one direction". Namely, if the program stops at step A3, the operation matrix contains at least the elements we want. However, it is difficult to prove that the program could always identify at step A5 a space which has two different representations. One then has to check in unclear cases the neighbouring elements of the diagram by hand. All the experiences thus far show that, if the program stops in A3, the resulting lattice is the right one.

REFERENCES

[1] Birkhoff, G.: Lattice Theory, American Mathematical Society Colloquium Publications, vol. XXV, 1973.

[2] Gross, H.: Quadratic Forms in Infinite Dimensional Vector Spaces, Birkhäuser, Boston 1979.

[3] Haapasalo, L.: Von Vektorraumisometrien induzierte Verbandsisomorphismen zwischen nicht orthostabilen und nicht distributiven Vektorraumverbänden. Universität Jyväskylä, Dissertation.

Manuskripteingang: 19. 6. 1980

VERFASSER:

Lenni Haapasalo und Pekka Niemistö, Mathematisches Institut der Universität Jyväskylä, Finnland