International Journal of Advanced Research and Development
ISSN: 2455-4030, Impact Factor: RJIF 5.24
www.advancedjournal.com
Volume 2; Issue 4; July 2017; Page No. 350-352

Role of C Algebra in lattice theory

Sarita Devi

Assistant Professor, Department of Mathematic, University College Kurukshetra, Haryana, India

Abstract
Lattice theory not only a necessary tool in the development of modern algebra in general and Universal algebra in particular. Lattice theory plays a major role in simplifying, unifying and generalizing many aspects of Mathematics and resembles Group theory, General topology and Functional analysis, because its central concept that of order, inter wines through almost all mathematics. The beauty of Lattice theory is in its extreme simplicity of the basic concept, which is order or partial order, one gets interesting generalization of lattice concept by dropping one or more of the lattice identities.

Keywords: lattice, C algebra, bounded

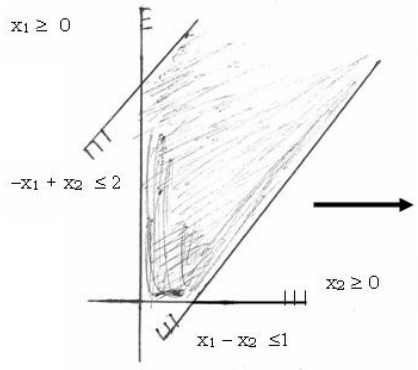
1. Introduction
In 1854, George Boole (1815–1864) introduced an important class of algebraic structures in his research work on mathematical logic. In his honor these structures have been called Boolean algebras. These are special type of lattices. In particular, congruence lattices play an important role. It was E. Schroder, who about 1890, considered the lattice concept in today's sense.
At approximately the same time, R. Dedikind developed a similar concept in his work on groups and ideals. Dedikind defined modular and distributive lattices which are types of lattices of that are important in applications. The rapid development of lattice theory started around 1930. We could say that Boolean lattices or Boolean algebras are the simplest and at the same time the most important lattices for applications.

Lattice Theory
It was Garrett Birkhoff's work in the mid thirties that started the general development of lattice theory. In a Brilliant series of papers he demonstrated the importance of the lattice theory and showed that it ii provides a unifying framework for previously unrelated developments in any mathematical disciplines.
During a pivot step, we make the value of a nonbasic variable just large enough to get the value of a basic variable down to zero. This, however, might never happen.
If we now try to bring x_2 into the basis by increasing its value, we notice that none of the tableau equations puts a limit on the increment. We can make x_2 and z arbitrarily large the problem is unbounded.
By letting x_2 go to infinity we get a feasible halfline - starting from the current BFS - as a witness for the unboundedness. In our case this is the set of feasible solutions
Example

Maximize x_1
Subject to $x_1 - x_2 \leq 1$,
$-x_1 + x_2 \leq 2$,
$x_1, x_2 \geq 0$.

with initial tablean

$$x_3 = 1 - x_1 + x_2$$
$$x_4 = 2 + x_1 - x_2$$
$$z = x_1$$



After one pivot step with x_1 entering the basis we get the tableau:

$$x_1 = 1 + x_2 - x_3$$
$$x_4 = 3 - x_3$$
$$z = 1 + x_2 - x_3$$

$\{(1,0,0,3) + x_2 (1, 1, 0, 0) | x_2 \geq 0\}$.

Such a halfline will typically be the output of the algorithm in the unbounded case. Thus, unboundedness can quite naturally be handled with the existing machinery. In the geometric interpretation it just means that the feasible polyhedron P is unbounded in the optimization direction.

While, we can make some nonbasic variable arbitrarily large in the unbounded case, just the other extreme happens in the degenerate case: some tableau equation limits the increment to zero so that no progress in z is possible.

The only candidate for entering the basis is x_2, but the first tableau equation shows that its value cannot be increased without making x_3 negative. This may happen whenever in a BFS some basic variables assume zero value, and such a situation is called degenerate. Unfortunately, the impossibility of making progress in this case does not imply optimality, so we have to perform a `zero progress' pivot step. In our example, bringing x_2 into the basis results in another degenerate tableau with the same BFS.

$$\begin{aligned} x_2 &= x_1 - x_3 \\ x_4 &= 2 - x_1 \\ \hline z &= x_1 - x_3 \end{aligned}$$

Nevertheless, the situation has improved. The nonbasic variable x_1 can be increased now, and by entering it into the basis, we already obtain the final tableau

$$\begin{aligned} x_1 &= 2 - x_4 \\ x_2 &= 2 - x_3 - x_4 \\ \hline z &= 2 - x_3 - x_4 \end{aligned}$$

With optimal BFS $x = (x_1 \ldots\ldots x_4) = (2,2,0,0)$

In this example, after one degenerate pivot we were able to make progress again. In general, there might be longer runs of degenerate pivots. Even worse, it may happen that a tableau repeats itself during a sequence of degenerate pivots, so the algorithm can go through an infinite sequence of tableaus without ever making progress. This phenomenon is known as cycling, and an example can be found. If the algorithm does not terminate, it must cycle. This follows from the fact that there are only finitely many different tableaus.

$$\begin{aligned} x_B &= \beta - \Lambda x_N \\ \hline z &= z_0 + \gamma^T x_N, \end{aligned}$$

and assume there is another tableau T_0 with the same basic and nonbasic variables, i.e. T is the system

$$\begin{aligned} x_B &= \beta' - \Lambda' x_N \\ \hline z &= z_0' + \gamma'^T x_N, \end{aligned}$$

By the tableau properties, both systems have the same set of solutions. Therefore

$$\begin{aligned} &(\beta - \beta') - (\Lambda - \Lambda') x_{N'} = 0 \text{ and} \\ &(z_0 - z_0') + (\gamma^T - \gamma'^T) x_N = 0 \end{aligned}$$

must hold for all d-vectors x_N, and this implies

$$\beta = \beta', \Lambda = \Lambda', \gamma = \gamma' \text{ and } z_0 = z_0'$$

Hence

T = T'

There are two standard ways to avoid cycling:

- Bland's smallest subscript rule: If there is more than one candidate x_k for entering the basis or more than one candidate for leaving the basis, which is another manifestation of degeneracy, choose the one with smallest subscript k.
- Avoid degeneracies altogether by symbolic perturbation. By Bland's rule, there is always a way of escaping from a sequence of degenerate pivots.

For this, however, one has to give up the freedom of choosing the entering variable. For us it will be crucial not to restrict the choice of the entering variable, so we will abandon Bland's rule and instead resort to the method of symbolic perturbation, although this requires more computational effort.

References

1. Borceux, F. Handbook of categorical algebra volume I, II, III, Cambridge University Press, 2011.
2. Bland, T. About Stone's notion of Spectrum, Journal of Pure and Applied Algebra, volume 197, 2010.
3. Coquand, T.; Spitters, B. Formal topology and constructive mathematics: the Gelfand and StoneYosida representation theorems, Journal of Universal Mathematics, 2009.
4. Dedikind. An elementary constructive proof of Gelfand duality for C*-algebras, submitted for publication, 2012.
5. Garrett Birkhoff, the categorical analysis of logic, Elsevier Science Publishers, 2011.
6. E. Schroder. Algebraic Geometry (8th edition), Springer-Verlag, 2010.
7. George Boole. Abstract Polytopes. PhD Thesis, Department of Operations Research, Stanford University, Stanford, California.
8. I. Adler and N. Megiddo. A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. J. ACM.
9. Adler and R. Saigal. Long Monotone Paths in Abstract Polytopes. Mathematics of Operations Research.
10. N. Amenta. Helly Theorems and Generalized Linear Programming. PhD thesis, University of California at Berkeley.
11. D. Avis and V. Chvatal. Notes on Bland's pivoting rule. Math. Programming Study, 8:24{3. B. Bixby. Personal communication.
12. J. Blomer. Computing sums of radicals in polynomial time. In Proceedings of 32nd Annual Symposium on Foundations of Computer Science.
13. J. Blomer. How to denest Ramanujan's nested radicals. In Proceedings of 33rd Annual Symposium on Foundations of Computer Science.
14. K. H. Borgwardt. The Simplex Method-A Probabilistic Analysis, volume 1 of Algorithms and Combinatorics, Springer-Verlag, Berlin Heidelberg, New York.
15. V. Chvatal. Linear Programming. W. H. Freeman, New York. 2013