Drawing Posets Using Less Ink

A. Aeschlimann* J. Schmid†

21. January 1992

1 Introduction

Defining aesthetics is hard, if not impossible, even in the restricted context of drawing diagrams of ordered sets – personal preferences play too prominent a rôle. There are quite a few algorithms around designed to draw such diagrams, either presenting a final solution without human interaction or, more modestly, helping the user to approximate interactively what he considers to be a "good " diagram. The first such program we are aware of is [FJ69] (of the interactive type), later improved and amplified in [JL79]. An algorithm operating without its users interference is described in [JG86]. A newer entry is the ANACONDA package used by the Wille school in their investigations into formal concept analysis, see, e.g. [KW87]. It proposes a raw solution which is improved by eye measure, using facilities to move points, enlarge picture size a. s. o.

Universally accepted criteria for a "good" diagram are that (i) it should have few intersections of edges at non-vertex points and (ii) the number of different slopes of edges should be small. On the other hand, every worker in the field has a mental list of dozens of diagrams which he would like to see drawn more or less his way by any algorithm he uses, regardless of abstract criteria. What we describe in this report is the use of two heuristic principles which, even to our own surprise, have led to quite usable diagrams for a wide range of posets.

These two principles are: (i) The total length of all edges of the diagram should be small (with the vertices kept at a minimal distance) and (ii) the vertices are constrained to coincide with the grid points of a given rectangular planar grid. The benefits are quite straightforward since (i) using less ink means less confusion and (ii) the restriction to grid points tends to keep the number of different slopes small (and the vertices at a safe distance, as required for the intended effects of (i)). The interactive part consists exclusively in correcting manually a possible side-effect which, however, does not occur in many situations. Ideas similar to these principles have surfaced in different contexts (see [RT81], [BC87], [B89], [T87], [SR83]) but have not been used to draw posets as far as we know.

This report is aimed to be quite self-contained. We describe the algorithm in raw form and exhibit some diagrams obtained in this way. Since the program was conceived as a readily usable tool (with the emphasis on results rather than on perfection), we are well aware of the fact that it will lend itself easily to improvements in many aspects.

*Address: IAM, Länggasstr. 51, 3012 Bern, aesch@iam.unibe.ch
†Address: MI, Sidlerstr. 5, 3012 Bern, schmid@mai.unibe.ch

The basic setup is to treat (vertical) Y-coordinates and (horizontal) X-coordinates of the vertices in two separate stages; starting with some reasonable values and improving them iteratively within the constraints imposed by our heuristics. Throughout, P is a finite poset with elements $x, y, \ldots$, which are assigned coordinates (X, Y) with respect to a rectangular grid.

It is mandatory at this point to stress the fact that there is a vast amount of work devoted to the study of diagrams of posets in a very broad sense. The canonical point of entry to this literature certainly is [R85]; concerning the rôles of slopes and levels, see e.g. [CP87], [C91] and [PR91].

2 Y-coordinates

Each iteration cycle consists of three basic components, i.e. an initialization, an iteration step, and a halting condition. The result of the initialization is given the iteration index $I = 0$. We write $y \prec\!\!\!- x$ to indicate that y is a lower neighbor of x.

Initialization We start with the Y-coordinates of a diagram closed downward, the dual of a diagram closed upward [K79], which defines the Y-coordinates in the following way:

$$\begin{aligned} Y(x)^0 &= 1, \text{ iff } \{y \mid y \prec\!\!\!- y\} = \emptyset \text{ and} \\ Y(x)^0 &= \max\{Y(y)^0 \mid y \prec\!\!\!- x\} + 1 \end{aligned}$$

Iteration Step Consider the next figure. It goes without saying that the total length of connections in the left diagram is smaller.

Let $n_u(x) = \text{card}\{y; y \succ\!\!\!- x\}$ and $n_l(x) = \text{card}\{y; y \prec\!\!\!- x\}$. If $n_u(x) > n_l(x)$, then x is pushed up as much as possible, i.e. $Y(x)^{I+1} := \min\{Y(y)^I | x \prec\!\!\!- y\} - 1$. If $n_l(x) = n_u(x)$, then $Y(x)^{I+1} := \lceil \frac{1}{2}(min\{Y(y)^I | x \prec\!\!\!- y\} + \max\{Y(y)^I | y \prec\!\!\!- x\}) \rceil$, a value roughly in the center.

Halting condition The procedure is stopped when no elements in the poset P are moved, i.e. $Y(x)^{I+1} = Y(x)^I$ for all $x \in P$. **The Y-coordinates are kept fixed from now on.**

Remark This procedure usually will not produce an optimal solution.

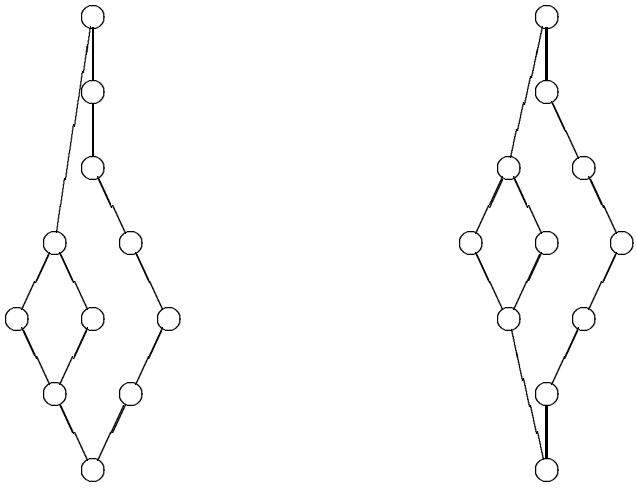

Figure 2.1 *The procedure will lead to a diagram of the type exhibited on the left side, while the total length of connections in the right diagram is shorter. (The X-coordinates in both diagrams are chosen such that the total length of connections is minimal with respect to the given Y-coordinates.)*

3 X-coordinates

The X-coordinates are now determined using the Y-coordinates. We denote a set of elements with same Y-coordinate as a *level*.

Initialization One of the levels with the maximal number of elements is taken to be the first whose elements obtain coordinates X^0. For each element x of this level l we compute a number $sum_l(x)$ which conveys the information how far away from the center of this level x should be put. This number is defined by

$$\begin{aligned} sum_l(x) &:= \sum_{y\in l} d\text{-}up_l(x,y) + \sum_{y\in l} d\text{-}down_l(x,y) && \text{where} \\ d\text{-}up_l(x,y) &:= \min\{Y(z) | x < z > y\} - Y(x), && \text{and} \\ d\text{-}down_l(x,y) &:= Y(x) - \max\{Y(z) | x > z < y\}. \end{aligned}$$

We now alternatively put an element on the left and right half, moving from the border towards the center of the level. The actual element x to be placed next is chosen by the following criteria (applied in the given order)

1. the highest score of $sum_l(x)$ (indicating that x may be put rather far away from the center),
2. the lowest values of $d\text{-}up_l(x,y)$, $d\text{-}down_l(x,y)$ with respect to the elements y that already are on the same half of the level,
3. the highest values of $d\text{-}up_l(x,y)$, $d\text{-}down_l(x,y)$ with respect to the elements y that already are on the opposite half of the level,

4. the lowest or highest number in some given enumeration of the elements for the left and right halves, respectively.

As soon as there is only one element left in the set of candidates, this one is taken, and the rest of the criteria is not considered. When all elements are placed, the basic distance of two consecutive elements is set to a value of 2 units.

The initial X-coordinates of the elements in the next higher level may now be determined as mean values of the X-coordinates of lower covers that are already known. If an element has no lower covers with known coordinates yet, the computation of its X-coordinate is postponed. The elements of the level under consideration are then pushed apart such that they acquire integer X-coordinates with a minimal horizontal distance of 2. This step is repeated for the level just below the initial one, and then iterated in the obvious way until no more elements get assigned X-coordinates. If there are elements left with no X-coordinate, there must exist two or more unconnected components in P. Select such an element and assign it a X-coordinate outside the range of values already used, and repeat until each element has a X-coordinate.

Remarks

- If x and y lie in unconnected components, or if there is no common upper or lower cover, then the corresponding value in *d-up*$_l$, *d-down*$_l$ is simply set to a "big" value, e.g. 1000.
- The original intention of *d-up*$_l$, *d-down*$_l$ is to provide some sort of *distance* between the elements of l. Since the actual definition does not fulfill the triangle inequality, one may be tempted to replace this definition by e.g. considering shortest paths in the underlying graph. However, if the poset is a lattice, then $\max\{Y(z)|x > z < y\}$ and $\min\{Y(z)|x < z > y\}$ coincide with $Y(x \wedge y)$ and $Y(x \vee y)$, respectively. If the operations $\wedge$ and $\vee$ are stored in tables, then the computation for *d-up*$_l$, *d-down*$_l$ may be considerably faster with $\wedge$ and $\vee$ than with shortest paths.

Iteration step Each iteration consist of two parts: Firstly, we move each element x horizontally in order to decrease the (local) sum of the distances between x and its upper and lower covers. Secondly we rearrange the level such that the coordinates are again integer with a minimal horizontal distance of 2 between each pair of adjacent elements within one level.

In the first part, we compute for each element x a value

$$X^J_{good}(x) := \frac{\sum_{y \prec x \text{ or } x \prec y} X(y)^J \cdot \frac{1}{|Y(y)-Y(x)|}}{\sum_{y \prec x \text{ or } x \prec y} \frac{1}{|Y(y)-Y(x)|}},$$

where $\frac{1}{|Y(y)-Y(x)|}$ is our choice of a possible weight for the X-coordinates of upper and lower covers (others are conceivable). Then we move x a bit towards this $X^J_{good}(x)$. The amount of shifting is determined by the number of upper and lower covers of x, i.e.

$$X^J_{dragged}(x) := X^J(x) + (1 - 2^{-(n_u(x)+n_l(x))}) \cdot (X^J_{good}(x) - X^J(x))$$

In the second part, we scan through each level from left to right, and identify *clusters* defined by the coordinates $X^J_{dragged}$: A cluster is a maximal subset C of a level l, satisfying

$$breadth(C) := \max\{X^J_{dragged}(x) | x \in C\} - \min\{X^J_{dragged}(x) | x \in C\} \leq 2(\,|C\,| - 1).$$

The elements in each cluster C are then pushed apart such that (i) the mean of all X-coordinates of its members remains invariant and (ii) the minimal distance between each pair of elements is 2 again. These coordinates are then rounded to integers, yielding the next set of coordinates $X|^{l+1}$.

Remark In theory, $X^J_{good}(x)$ is only a moderate substitute for $X^J_{eff}(x)$, the value which really minimizes the local distances

$$\sum_{y \prec x \text{ or } x \prec y} \sqrt{(X(x) - X(y))^2 + (Y(x) - Y(y))^2}$$

where $X(x)$ is the argument. Finding $X^J_{eff}(x)$ means finding a solution to

$$\frac{d}{dX(x)} \sum_{y \prec x \text{ or } x \prec y} \sqrt{(X(x) - X(y))^2 + (Y(x) - Y(y))^2} =$$

$$\sum_{y \prec x \text{ or } x \prec y} \frac{X(x) - X(y)}{\sqrt{(X(x) - X(y))^2 + (Y(x) - Y(y))^2}} = 0$$

using e.g. approximative methods. The benefits of this approach are unknown to us, but not considered too overwhelming, while computation time considerably increases.

Halting condition After each iteration, the total length tl of the distances is computed and compared to tl of the last iteration. The iteration is terminated as soon as tl decreases by less than a previously given $\epsilon > 0$ (or even increases).

Tuning the method We do not know how close our results are to a diagram with minimal total length of edges. There are several heuristics and weight functions that may be fine-tuned in order to produce different diagrams.

Remark It may happen that an edge crossing a level passes through an element within this level. This phenomenon is not desired since it may mislead to the assumption that the element in question might be an upper or lower cover of the edge's endpoints. Furthermore, any incidental edge connected to this element and having the same slope as the first one is not shown, since the two lines coincide. We correct this phenomenon manually (see the edge from $\acute{E}$ to A_2 in fig. 5.2). Other possible solutions might be:

- Move the vertex and the endpoints of the edge in question only marginally such that the local structure of the poset becomes visible.
- Represent the edges by straight line segments or splines. Calculate the endpoints of the line segments or the intermediate points of the spline such that the line passes neither through elements nor through eventual labels thereof.

- If one is not willing to allow any loss of aesthetics, then there is the possibility to add some more constraints using "dummy vertices". For each time an edge crosses a level, a dummy vertex on this edge is defined. The constraints can then be formulated in a straightforward way: (i) all dummy vertices of an edge must be collinear, (ii) a dummy vertex should have some predefined minimal distance to the next element on that level, and (iii) two dummy vertices on the same level may coincide. It is open how these constraints can be taken into account with our algorithm. Further pursuit of this question might be worthwhile.

4 Technicalities

The program is written in TURBO-PASCAL and runs on any compatible PC. The input consists mainly of a description of the covering relation of the poset. The current version can handle posets with up to 255 elements. The output consists of a file with the coordinates added to the covering relation. There exists some ad hoc software to convert this output into a LaTeX `\picture`, which can be included in any LaTeX document. The large pictures were drawn using an IBM mainframe with a VERSATEC plotter. The drawing software there is written in PASCAL/VS and uses FORTRAN subroutines to control the plotter.

5 Some samples

What follows is a loose collection of some diagrams, obtained by the procedures sketched above.

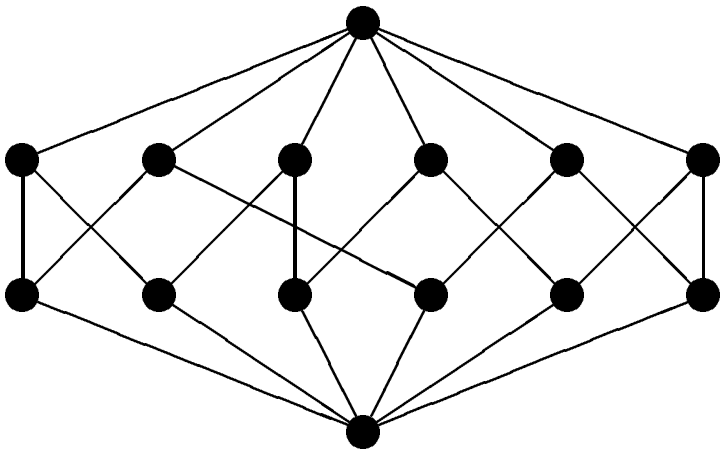

Figure 5.1 *Rival's crown, augmented with a top and a bottom element.*

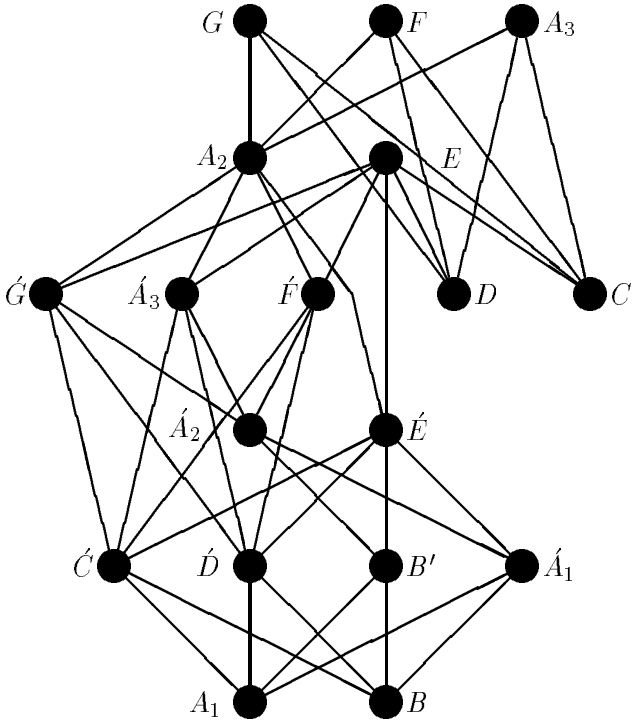

Figure 5.2 *Titlepage of ORDER, Vol. 7, No. 4, 1990/1991. The total length of edges is about 14% smaller than in the original.*

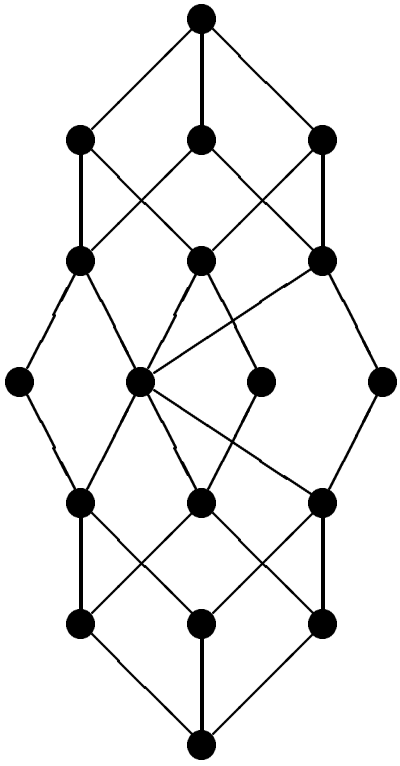

Figure 5.3 *Free distributive lattice on three generators FD(3)*

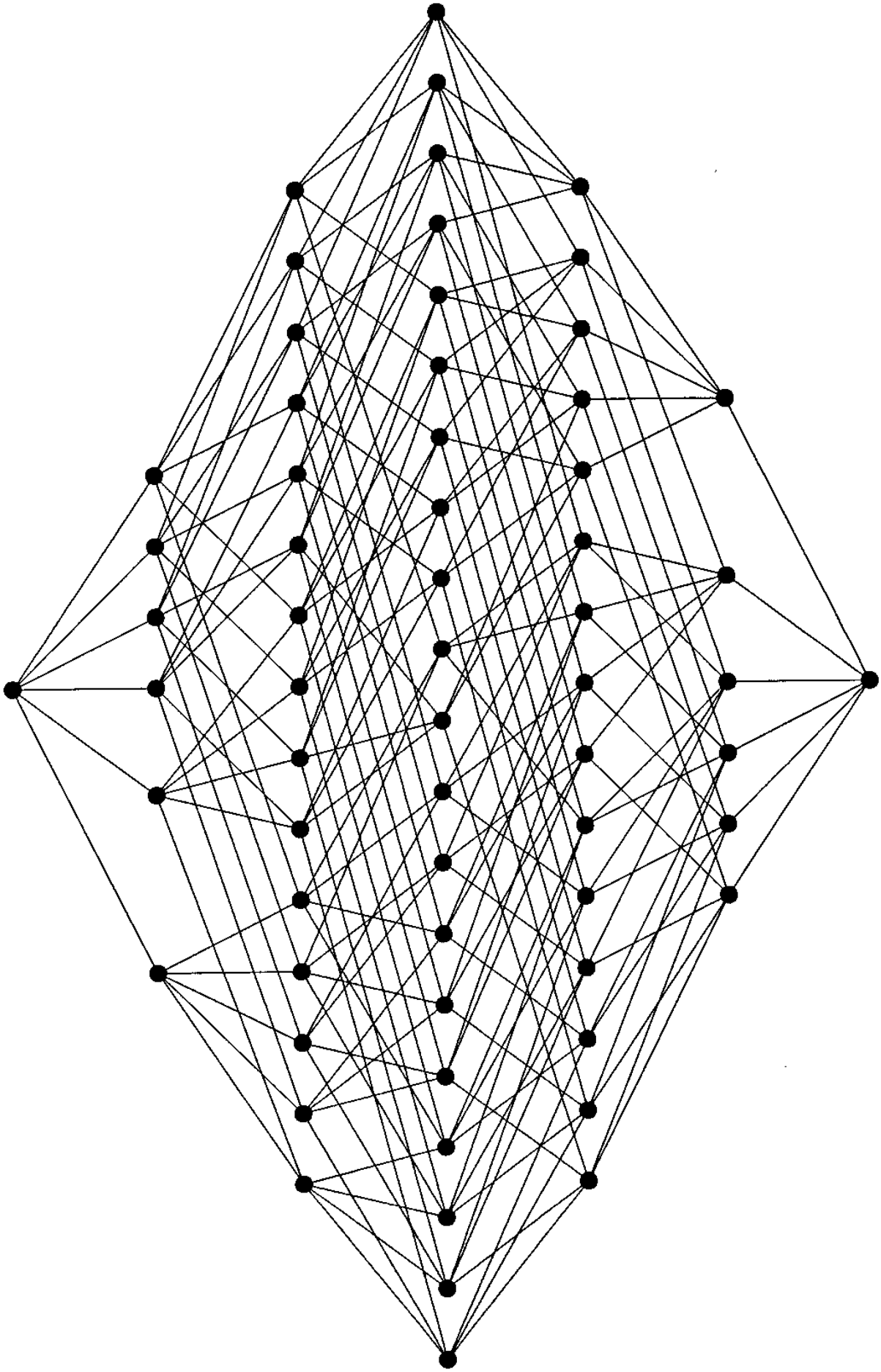

Figure 5.4 *Previous page: Boolean lattice of order 6*

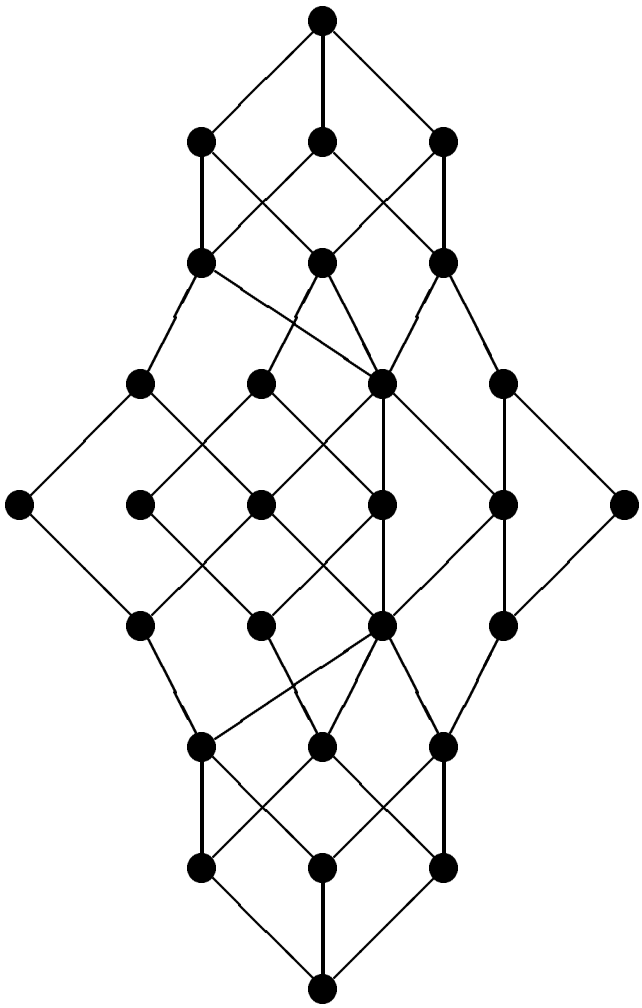

Figure 5.5 *Free modular lattice on three generators FM(3)*

The boolean lattice of order 8 doesn't fit onto a single page. Anyone whishing to see its diagram as produced by our algorithm is kindly invited to inspect the office door of this journals editor (for US/Canada residents) or to contact the authors for a copy (all other countries).

References

[B89] Brandenburg, F. J.: *On the Complexity of Optimal Drawings of Graphs*, Preprint, Univ. Passau, 1989.

[BC87] Bhatt, S. N., Cosmadakis, S. S.: *The Complexity of Minimizing Wire Lengths in VLSI Layouts*, Inf. Process. Letters **25** (1987) pp. 263 – 267.

[C91] Czyzowicz, J., *Lattice Diagrams with Few Slopes*, J. Comb. Theory Series A **56** (1991), pp. 96 – 108.

[CP87] Czyzowicz J., Pelc A., Rival I., Urrutia J.: *Crooked Diagrams with Few Slopes*, Tech. Rep. TR-87-26 (1987) University of Ottawa.

[FJ69] Ferber, K., Jürgensen, H.: *A Programme for the Drawing of Lattices*, in: J. Leech (ed.), Computational problems in abstract algebra, Oxford, 1969, pp. 83 – 87.

[JG86] Janes R., Gaskill H.: *An Algorithm for the Generation and Display of Finite Geometric Lattices*, Congressus Numerantium **52** (1986). pp. 125 – 145.

[JL79] Jürgensen, H., Loewer, J.: *Drawing Hasse Diagrams of Partially Ordered Sets*, Preprint, 1979.

[KW87] Kipke U., Wille R..: *Formale Begriffsanalyse erläutert an einem Wortfeld*, Preprint Nr. 1046, TH Darmstadt, 1987.

[K79] Kyuno S.: *An Inductive Algorithm to Construct Finite Lattices*, Math. Comp. Vol. 33, (1979), pp. 409 – 421.

[PR91] Pelc A., Rival I.: *Orders with Level Diagrams*, Eur. J. Comb. **12** (1991) pp. 61 – 68.

[RT81] Reingold E. M., Tilford J. S.: *Tidier Drawings of Trees*, IEEE Trans. on SW Eng. SE-7, 2, March 1981, pp. 223 – 228.

[R85] Rival I.: *The Diagram*, in: Graphs and Orders, Reidel Publ. Co. Dordrecht 1985, pp. 103 – 133.

[SR83] Supovit K. J., Reingold E. M.: *The Complexity of Drawing Trees Nicely*, Acta Informatica **18**, (1983), pp. 377 – 392.

[T87] Tamassia, R.: *On Embedding a Graph in the Grid with the Minimum Number of Bends*, SIAM J. Comput. **16** (1987), pp. 421 – 444.