Partially Ordered Sets and Lattices

Domingo López Rodríguez

Practice

Estructuras Algebraicas

Practice to be carried out with R. Follow the instructions and complete the questions that are requested, delivering them in PDF format. It is recommended to use the Rmarkdown or Quarto format from RStudio to generate a document that answers the questions. The answers to the questions must be properly reasoned (it is not enough just to execute code and show the answer produced by the software). Care should be taken in presentation and proper written expression.

Contents

Introduction

In this practice, we will use an R library that is hosted on GitHub. To install it, the following commands must be executed in R's *prompt*:

```
install.packages("remotes")
remotes::install_github("malaga-fca-group/LatticeTheory")
```

> You will only be able to go ahead in this practice if you manage to install this library, since the rest of the library will make use of the methods and objects defined in it.

In order to load the library into memory, and thus be able to make use of its functions, the following command must be executed:

```
library(LatticeTheory)
```

In this document, we will briefly review the theory with examples, while working on these examples with R. The definitions will have a blue background, the annotations with a yellow or red background, and the proposed exercises with a grey background.

Posets

Recall that a poset (short for **p**artial **o**rdered **set**) is a set X endowed with a (partial) order relation $\leq$ which, by definition, verifies the following properties:

- Reflexive: $x \leq x$ for all $x \in X$.
- Antisymmetric: if $x, y \in X$ verify that $x \leq y$ and $y \leq x$, then $x = y$.
- Transitive: if $x, y, z \in X$ verify $x \leq y$ and $y \leq z$, then we have that $x \leq z$.

If, in addition, it is verified that, for all $x, y \in X$, it is either $x \leq y$ or $y \leq x$, then it is called **total order**.

A *computational* way of expressing an order relation is by means of a square matrix A, where each row and column represents an element of X and the values of $a_{i,j}$ are: 1, if the j-th element of X is less than or equal to the i element; 0 otherwise. For instance:

```
  a b c d e
a 1 0 0 0 0
b 1 1 0 0 0
c 1 1 1 0 0
d 1 0 0 1 0
e 1 1 1 1 1
```

In this example, $X = \{a, b, c, d, e\}$ and we know that, for example, $b \leq c$ and $c \leq e$.

Construction

In order to construct a *poset* using this library, it is first necessary to define the relation matrix.

```
A <- matrix(
  c(
    1, 0, 0, 0, 0,
    1, 1, 0, 0, 0,
```

```
    1, 1, 1, 0, 0,
    1, 0, 0, 1, 0,
    1, 1, 1, 1, 1
  ),
  nrow = 5, # Number of rows
  ncol = 5, # Number of columns
  byrow = TRUE, # Values are provided by row
  dimnames = list(
    c("a", "b", "c", "d", "e"),
    c("a", "b", "c", "d", "e")
  ) # Elements names
)
```

Once the order matrix has been defined, we can create a `Poset` object as follows:

```
my_poset <- Poset$new(A)
```

> The elements of the set X take their names from the names of the rows and columns of the matrix we pass to it.

Once we have this object, we can verify its contents:

```
my_poset
```

```
Poset with 5 elements.
Elements: a, b, c, d, e.
Strict comparabilities:
a < b, a < c, a < d, a < e
b < c, b < e
c < e
d < e
```

Most useful functions

From now on, we will see operators and methods defined on objects of class `Poset` that reproduce the most important operations seen in theory.

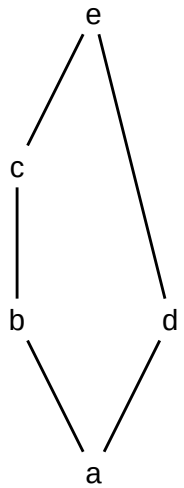
Representation of the Hasse diagram

The most visual function of all that we will present, as it is dedicated to graphically representing the Hasse diagram of a *poset*.

> The **Hasse diagram** of $(X, \leq)$ is the graph-like scheme (X, E), where each node represents an element of X, and the edges correspond to the direct order relation: there is an edge between a and b if and only if $a \leq b$ and there is no element "in between": there is no c such that $a \leq c \leq b$.
> The graphical representation orders the nodes from bottom to top following the order used in the poset, so that if $a \leq b$, then the node a will be below b in the graph.

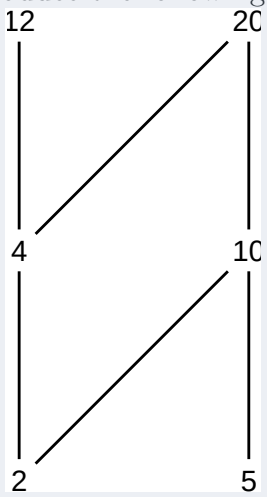
The function to apply is `plot()`:

```
my_poset$plot()
```



Exercise 1

Write the code necessary to reproduce the following Hasse diagram:



Notable elements

The functions we will see next will serve to determine the most notable elements within a subset Y of X.

Maximal and minimal elements

> Given $Y \subseteq X$, an element a is **maximal** if there is no $b \in Y$, $b \neq a$, such that $a \leq b$. In other words, if there is no element above it according to the defined order. Similarly, a **minimal** element is defined.

The functions to calculate such elements are `maximals()` and `minimals()`. For example,

```
my_poset$maximals("a", "b", "d")
```

```
[1] "b" "d"
```

```
my_poset$minimals("b", "c", "d")
```

```
[1] "b" "d"
```

Note: Notice that the names of the elements are strings, so they must be enclosed in quotation marks.

Exercise 2

From the poset of the previous exercise, calculate the maximal and minimal elements of $Y = \{2, 4, 12, 20\}$.

Maximum and minimum

Given $Y \subseteq X$, we will say that $y \in Y$ is the **maximum** element if $x \leq y$ for every $x \in Y$, that is, if y is greater than or equal to every element of Y.
Similarly, the **minimum** element is defined.

To calculate them, we have the methods `maximum()` and `minimum()`. :

```
my_poset$maximum("a", "b", "c")
```

```
[1] "c"
```

```
my_poset$maximum("a", "b", "d")
```

```
NULL
```

```
my_poset$minimum("a", "b", "d")
```

```
[1] "a"
```

```
my_poset$minimum("b", "c", "d")
```

```
NULL
```

As we can see, for a given Y, the maximum or minimum elements may not exist. There will always be maximal and minimal elements, but the maximum or minimum, by the given definition, must be unique.

Exercise 3

We continue with the poset of our exercises:

a) Calculate the maximum and minimum, if they exist, of $\{2, 4, 5, 20\}$.

b) Calculate the maximum and minimum, if they exist, of $\{2, 4, 12, 20\}$.

Upper and lower bounds

Given $Y \subseteq X$, an element $a \in X$ is an **upper bound** of Y if $y \leq a$ for every $y \in Y$. In other words, the upper bounds of Y are elements of X greater than all elements of Y.

Similarly, **lower bounds** are defined.

The methods for these bounds are `upper_bounds()` and `lower_bounds()`. For example:

```
my_poset$upper_bounds("a", "b", "d")
```

```
[1] "e"
```

```
my_poset$lower_bounds("b", "c", "d")
```

```
[1] "a"
```

Exercise 4

In the poset of our exercises:

a) Calculate the upper and lower bounds of the set $Y = \{4, 10\}$.

b) What are the upper and lower bounds of $\{4, 10, 12\}$?

c) Provide a set that has neither upper nor lower bounds.

d) Provide an example of a set Y that has upper and lower bounds but neither maximum nor minimum.

Supremum and infimum

The minimum of the upper bounds of a set $Y \subseteq X$, if it exists, is called the **supremum** (it is the smallest of the upper bounds).
The maximum of the lower bounds, if it exists, is called the **infimum**.

We calculate these elements, if they exist, using the functions `supremum()` and `infimum()`:

```
my_poset$supremum("b", "c", "d")
```

```
[1] "e"
```

```
my_poset$infimum("b", "c", "d")
```

```
[1] "a"
```

If the supremum of Y exists and belongs to Y, then it coincides with the maximum of Y. Similarly, if the infimum of Y exists and belongs to Y, then it coincides with the minimum of Y.
And conversely, if Y has a maximum, then that element is also the supremum of Y, and if it has a minimum, that element is, in turn, the infimum of Y.

Exercise 5

Provide an example, in the poset we are using in the questions, of:

a) A set Y_1 that has supremum and infimum.

b) A set Y_2 that has supremum but not maximum.

Linear extension

A **linear extension** of $(X, \leq)$ is a total order relation $\preceq$ also defined on X that *preserves the order* given by $\leq$, meaning that if $a \leq b$, then $a \preceq b$.

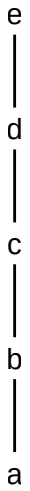
The method to calculate a linear extension (there can be many possible extensions) is `linear_extension()`, and it returns another `Poset`:

```
extension <- my_poset$linear_extension()
extension
```

```
Lattice with 5 elements.
Elements: a, b, c, d, e.
Strict comparabilities:
a < b, a < c, a < d, a < e
b < c, b < d, b < e
c < d, c < e
d < e
```

If we notice, this is equivalent to $a < b < c < d < e$, and we can verify that it is a total order if we draw its Hasse diagram:

```
extension$plot()
```



Lattices

A particular case of a partially ordered set is what we call a lattice.

A **lattice** is a poset $(L, \leq)$ where for each pair of elements $x, y \in L$, their supremum and infimum exist. We denote:

$$\sup\{x, y\} = x \vee y := \text{supremum of} \{x, y\}$$

$$\inf\{x, y\} = x \wedge y := \text{infimum of} \{x, y\}$$

A lattice is said to be **complete** if the supremum and infimum of any subset of elements exist.

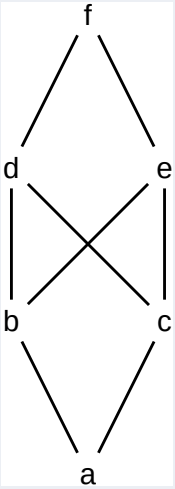
From an algebraic standpoint, we could define a lattice as a set L equipped with two operators $\vee$ and $\wedge$ that satisfy certain conditions. However, in this practice, we are interested in order relations, so we will overlook that other construction for now. Nevertheless, when convenient, we will use the notation $(L, \vee, \wedge)$ to denote the lattice equipped with those two operators.

Exercise 6

Does the poset we are using in the exercises have a lattice structure?

Exercise 7

Does the poset given by the following Hasse diagram have a lattice structure?



Construction

While an object of type `Lattice` can be constructed as a `Poset`, based on the matrix defining the partial order, there are two simple ways to construct it under special conditions:

a) If we already have an object of type `Poset`, we can apply the function `to_lattice()`, and *if it is a lattice*, it returns an object of type `Lattice`.

b) In the library, there are already certain types of predefined lattices that can help us work, which we will list in the following section.

Some examples of lattices

These predefined types always return objects of the class `Lattice`.

Chains

> A **chain** is a set equipped with a total order.

For instance:

```
my_chain <- chain(4) # Number of elements in the chain
my_chain
```

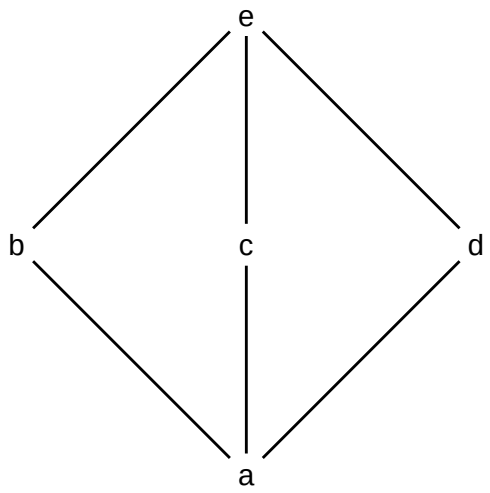
```
Lattice with 4 elements.
Elements: a, b, c, d.
Strict comparabilities:
a < b, a < c, a < d
b < c, b < d
c < d
```

```
my_chain$plot()
```

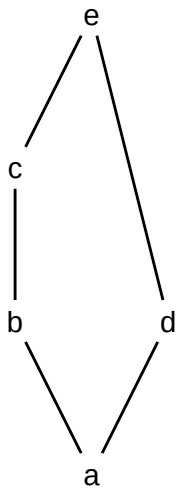
Diamond and pentagon

The easiest way to understand what these lattices are is to look at their Hasse diagrams:

```
D <- diamond()
D$plot()
```




```
P <- pentagon()
P$plot()
```



Divisors

The **lattice of divisors** of a positive integer n is denoted by $(\mathcal{D}_n, |)$, where

$$\mathcal{D}_n := \{m \in \mathbb{Z}^+ : m \text{ divides } n\}$$

and the partial order relation | is given by: $m|k$ if, and only if, m divides k.

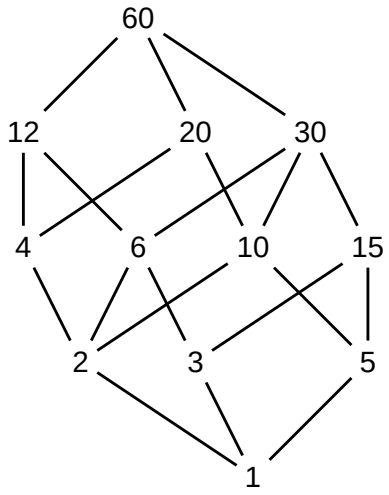
In this case, moreover, we can see that the supremum and infimum operators correspond, in fact, to the arithmetic operations we are familiar with from childhood: least common multiple (LCM) and greatest common divisor (GCD). In other words, $(\mathcal{D}_n, \text{LCM}, \text{GCD})$ is an *algebraic lattice*.

The syntax is `divisors_lattice(n)`. Here is an example:

```
div <- divisors_lattice(60)
div
```

```
Lattice with 12 elements.
Elements: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60.
Strict comparabilities:
1 < 2, 1 < 3, 1 < 4, 1 < 5, 1 < 6, 1 < 10, 1 < 12, 1 < 15, 1 < 20, 1 < 30, 1 < 60
2 < 4, 2 < 6, 2 < 10, 2 < 12, 2 < 20, 2 < 30, 2 < 60
3 < 6, 3 < 12, 3 < 15, 3 < 30, 3 < 60
4 < 12, 4 < 20, 4 < 60
5 < 10, 5 < 15, 5 < 20, 5 < 30, 5 < 60
6 < 12, 6 < 30, 6 < 60
10 < 20, 10 < 30, 10 < 60
12 < 60
15 < 30, 15 < 60
20 < 60
30 < 60
```

```
div$plot()
```



Exercise 8

Build the lattice of divisors of 90 and draw its Hasse diagram. This will be the lattice we will use in the rest of the exercises in this section.

Most useful functions

As it is a particular case of `Poset`, **all** the functions and methods mentioned in the Posets section remain valid. However, due to the more restricted nature of lattices, other notable elements and more specific situations can be defined, which we will see below.

Top and bottom

A lattice $(L, \leq)$ is called **bounded** if the maximum and minimum elements of L exist (equivalently, supremum and infimum of L). The supremum of L is usually called **top** and is denoted by $\top$ or by 1 (due to its algebraic properties, as it is the identity element of the infimum operator). The infimum of L is called **bottom** and is denoted by $\bot$ or 0, as it is the identity element of the supremum operator.

Note: Every finite lattice is complete and bounded.

The methods that return these elements of a lattice are `top()` and `bottom()`.

Exercise 9

What are the elements $\top$ and $\bot$ of the pentagon lattice? And of the diamond lattice? And of the lattice of divisors of 90?

Atoms and co-atoms

An **atom** in a bounded lattice $(L, \leq)$ is an immediate upper neighbor of the minimum element $\bot$.
Similarly, a **co-atom** is each of the immediate lower neighbors of the maximum element $\top$.

The corresponding functions are `atoms()` and `coatoms()`.

Exercise 10

Determine the atoms and coatoms of the lattice of divisors of 90.
What is the relationship of the atoms to the prime factors of 90?

Irreducible elements

An element $x \in L$ is called $\vee$**-irreducible** (supremum irreducible) if it cannot be written as the supremum of two distinct elements other than itself.
An element $x \in L$ is called $\wedge$**-irreducible** (infimum irreducible) if it cannot be written as the infimum of two distinct elements other than itself.

The importance of these elements lies in **Birkhoff's Representation Theorem**, which

tells us that any element of the lattice can be written (in a certain sense, uniquely) as the supremum of $\vee$-irreducible elements and as the infimum of $\wedge$-irreducible elements.

The simple way to determine these elements in the lattice, graphically, is as follows:

- $\vee$-irreducible elements are those that have only one immediate lower neighbor (only one element connected immediately below).
- $\wedge$-irreducible elements are those that have only one immediate upper neighbor.

The functions to use to determine these elements are `join_irreducibles()` for the $\vee$-irreducibles and `meet_irreducibles()` for the $\wedge$-irreducibles.

Exercise 11

Determine the irreducible elements of the lattice of divisors of 90.
What relationship exists between the $\vee$-irreducible elements and the prime factorization of 90?

Complement

A bounded lattice is called **complemented** if every element $x \in L$ has a complement, meaning there exists $\hat{x} \in L$ such that $x \vee \hat{x} = \top$ and $x \wedge \hat{x} = \bot$.

Note: The complement of an element may not exist, and if it exists, it may not be unique.

The method to determine the possible complements of a given element is `complements()`. For example, to calculate the complementary elements of the element d in the pentagon

represented above, we will do:

```
P <- pentagon()
P$complements("d")
```

```
[1] "b" "c"
```

> **Note**: Indeed, it is always true that the complement of $\top$ is $\bot$ and vice versa.

> **Exercise 12**
> Is there any element in the lattice of divisors of 90 that does not have a complement? And any element with more than one complement?

Boole algebras

A algebraic condition that can be studied on lattices is distributivity:

> A lattice $(L, \vee, \wedge)$ is said to be **distributive** if it satisfies the following distributive laws: for all $x, y, z \in L$, we have
>
> $$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
>
> $$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Examples of non-distributive lattices are the diamond and the pentagon, mentioned earlier. In fact, a very important theoretical result tells us when a lattice is distributive: A lattice is distributive if, and only if, it does not contain sublattices isomorphic to the diamond or the pentagon.

Furthermore, if a lattice is distributive, we can ensure that if an element has a comple-

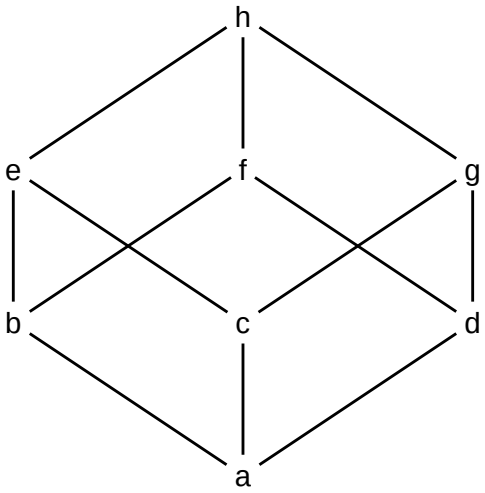
ment, it is unique.

It is interesting, therefore, to study complemented and distributive lattices, and they receive a specific name.

> A bounded, distributive and complemented lattice is called **Boolean algebra**.

To create a Boolean algebra with n atoms, the syntax `boole(n)` is used.

```
B <- boole(3)
B$plot()
```



Additional notes

In this practice, we have focused on operating and reasoning about lattices and partially ordered sets, and we have left aside other ideas such as sublattices or isomorphism of lattices (although at some point we have mentioned them in passing). To complete the theoretical part, it is essential to consider the material provided in class and available on the virtual campus.