Orthogonal Lattice Exploration

by Sven Nilsen, 2022

In this paper I present an algorithm that generates axis aligned extremes from finite lattice structures without knowing the internal representation of the lattice, but from random samples. This can be used to efficiently explore finite lattice structures using e.g. n-trees.

In a finite lattice structure[1], the "meet" and "join" of any two elements `a` and `b` converges toward elements at the "diagonal" of the lattice:

$a \wedge b$	Meet	e.g. minimum
$a \vee b$	Join	e.g. maximum

For example, in a cube, one can choose a diagonal from `(0, 0, 0)` to `(1, 1, 1)` such that meets and joins converges towards the diagonal. This is because the extreme minimum and extreme maximum lies at the diagonal.

Without knowing the dimensionality of the finite lattice structure, one can not construct directly the axis aligned extremes. For example, in a cube, `(1, 0, 0), (0, 1, 0), (0, 0, 1)` are these extremes.

To determine whether some element `c` lies within an axis aligned region bounded by `a, b`:

$$\text{inside}(a, b, c) := ((a \wedge b) \wedge c == (a \wedge b))\ \&\ ((a \vee b) \vee c == (a \vee b))$$

Now, assume that one can generate a random sample within any axis aligned region.

$$\text{inside}(a, b, \text{random\_element}(a, b))$$

There is an algorithm that can explore the lattice "orthogonally" to the diagonal.
It works when the following condition holds for `a, b`:

$$\text{explorative}(a, b) := ((a \wedge b)\ != a)\ \&\ ((a \wedge b)\ != b)$$

Let `c` be some random element bounded by `a, b` and `min, max` the extremes of the diagonal.
The element `c` is converging towards `a`'s corner when:

$$\text{explore}(a, b, c) := \text{inside}(b \wedge c, b \vee c, a)\ \&\ !\text{inside}(\min, a \vee b, c)\ \&\ !\text{inside}(a \wedge b, \max, c)$$

To get `b`'s corner, one can simply swap `a, b`.

When the orthogonal extremes have been found, one can use them to generate axis aligned regions that cover the domain of the finite lattice using some random sample as origo. This in turn can be used to efficiently explore finite lattice structures using e.g. n-trees.

One advantage of this algorithm is that it does not require any metric on the lattice and it works for any sub-lattice structure of infinite lattices. For example, recursive data structures with partial order.

References:

[1] “Lattice (order)”
Wikipedia
https://en.wikipedia.org/wiki/Lattice_(order)