Treillis modulaire et fondamentaux

Franck Jeannot

Montréal, Canada, Mai 2023, AA823, v1.0

Abstract

A basic review on the concepts of $\wedge$ for the glb "greatest lower bound" and $\vee$ for lub "least upper bound" in lattices.

Keywords: Treillis, glb, lub, intersection, jonction, lattice, couple modulaire

1. Introduction

Le symbole $\wedge$ est couramment utilisé pour représenter l'opération d'**intersection (ou meet)** dans le contexte des treillis [1] [2]. Il est utilisé pour décrire la plus grande borne inférieure (ou **glb**, pour greatest lower bound) de deux éléments dans un treillis.

En termes simples, le glb de deux éléments dans un treillis est l'élément le plus bas (ou le plus petit) qui est inférieur ou égal à ces deux éléments. C'est une notion fondamentale en théorie des treillis, où les éléments sont partiellement ordonnés et peuvent avoir des relations de dominance.

Le symbole $\wedge$ est inspiré du symbole utilisé en logique mathématique pour représenter la conjonction logique (et). Dans le contexte des treillis, il est utilisé pour indiquer l'opération de **rencontre**, qui est une généralisation de la conjonction logique aux éléments d'un treillis.

Ainsi, lorsque l'on écrit $(x \vee a) \wedge b = x$, cela signifie que l'intersection (glb) des éléments $x \vee a$ et b est égale à x. Cela exprime la relation de dominance dans le treillis où x est inférieur ou égal à $x \vee a$ et b, et il n'y a pas d'élément inférieur à x qui est plus grand que $x \vee a$ et b.

Le symbole (lub) est une abréviation de "least upper bound" en anglais, ce qui signifie la plus petite borne supérieure. Il est utilisé pour indiquer l'opération de jonction, qui est une généralisation de la disjonction logique aux éléments d'un treillis.

Ainsi, lorsque l'on parle du (lub) de deux éléments x et y dans un treillis, cela fait référence à l'élément le plus petit qui est supérieur ou égal à x et y. On peut le noter sous la forme $x \vee y$, où le symbole $\vee$ représente l'opération de jonction.

2. Couples modulaires

Dans un treillis, un **couple modulaire** est un couple d'éléments (a, b) qui satisfait la propriété de modularité. La propriété de modularité stipule que pour tout élément x tel que $a \wedge b \leq x \leq b$, l'égalité $(x \vee a) \wedge b = x$ est vérifiée.

Dans la formule mathématique : $(x \vee a) \wedge b = x$, on a :

- $\wedge$ représente l'opération d'intersection (glb) dans le treillis.
- $\vee$ représente l'opération d'union (lub) dans le treillis.
- $x \vee a$ représente le lub (plus petit élément supérieur) de x et a.
- $(x \vee a) \wedge b$ représente l'intersection (plus grand élément inférieur) de $(x \vee a)$ et b.

La propriété modulaire indique que lorsque cette égalité est satisfaite, l'opération d'intersection de $(x \vee a)$ et b donne simplement x. En d'autres termes, $(x \vee a)$ et b se "séparent" et récupèrent la valeur de x.

Cette propriété est importante en théorie des treillis et trouve des applications dans différents domaines, tels que l'algèbre booléenne, la théorie des ensembles, la théorie des ordres, etc. Elle permet de caractériser certains types de treillis et fournit des informations sur les relations entre les éléments du treillis.

3. Exemple d'application python

```python
# Lattice example
lattice = {
    'A': ['B', 'C', 'D'],
    'B': ['D'],
    'C': ['D'],
    'D': []
}

# Function to compute the greatest lower bound (glb)
def glb(a, b):
    common_elements = set(lattice[a]).intersection(set(lattice[b]))
    if common_elements:
        return min(common_elements)
    else:
        return None

 Function to compute the least upper bound (lub)
def lub(a, b):
    all_elements = set(lattice[a]).union(set(lattice[b]))
    if all_elements:
        return max(all_elements)
    else:
        return None

# Function to compute other essential lattice values
def compute_lattice_values():
    elements = lattice.keys()
    infimum = None
    supremum = None
```

```
    for element in elements:
        if not lattice[element]:
            if infimum is None:
                infimum = element
            else:
                infimum = glb(infimum, element)

        if set(lattice[element]) == set(elements):
            supremum = element

    return elements, infimum, supremum

# Main code
if __name__ == '__main__':
    elements, infimum, supremum = compute_lattice_values()

    print("Lattice elements:", elements)
    print("Infimum:", infimum)
    print("Supremum:", supremum)

    # Example glb and lub calculations
    x = 'A'
    y = 'C'
    print("glb({0}, {1}):".format(x, y), glb(x, y))
    print("lub({0}, {1}):".format(x, y), lub(x, y))
```

```
python .\lattice.py
Lattice elements: dict_keys(['A', 'B', 'C', 'D'])
Infimum: D
Supremum: None
glb(A, C): D
lub(A, C): D
```

Références

[1] Franck Jeannot, Contrôles d'accès basés sur treillis et modèles de sécurité, u638 (12 2019).
URL https://franckybox.com/wp-content/uploads/Lattice_security_models.pdf

[2] Treillis modulaire.
URL https://fr.wikipedia.org/wiki/Treillis_modulaire