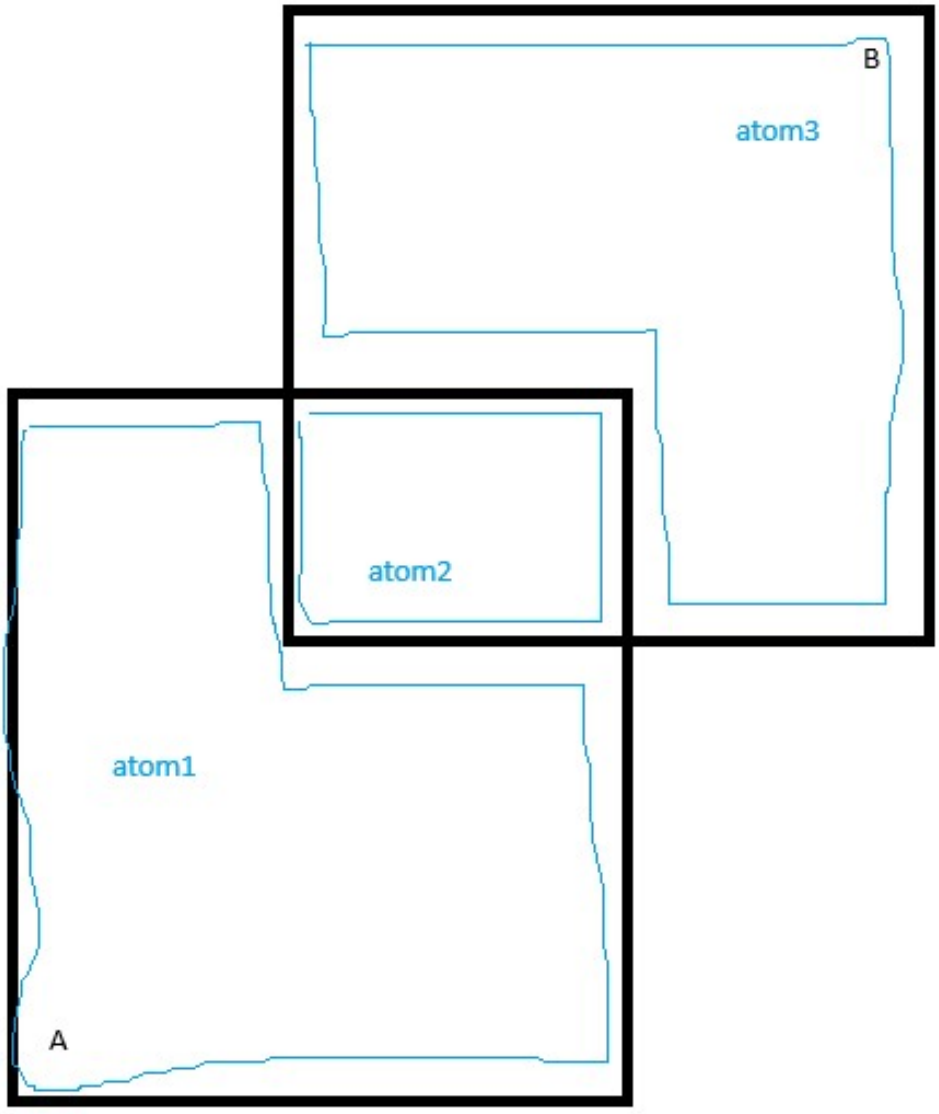

B
atom3
atom2
atom1
A

REULT

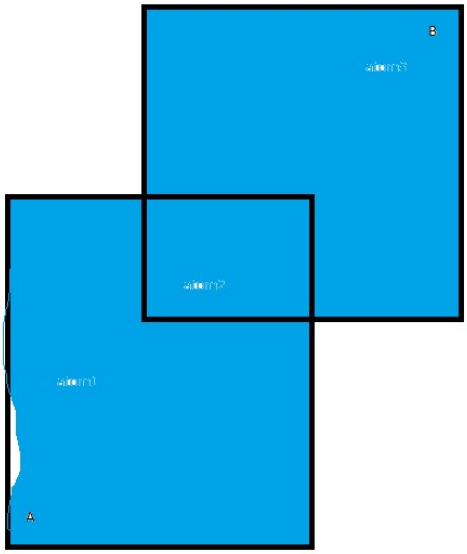
UNION(A,B)

	A	B	Result
Atom1	1	0	**1**
Atom2	1	1	**1**
Atom3	0	1	**1**

```
function Union(v::AbstractArray)
  return any(v)
en

@assert(Union([false, false])==false)
@assert(Union([true,  false])==true)
@assert(Union([true,  true])==true)
@assert(Union([false, true])==true)
```

NOTE: Result is needed to know if an atom will be included in the boolean operation, or must be excluded

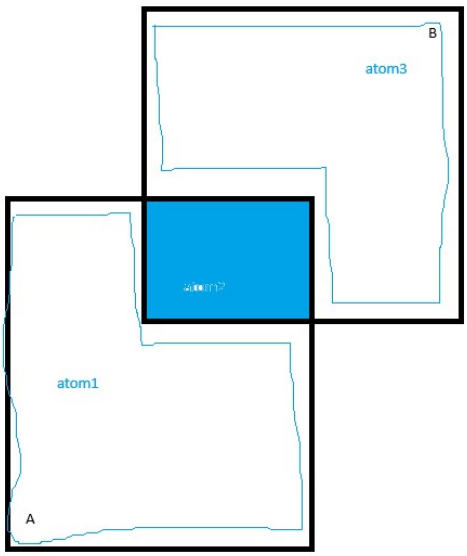


INTERSECTION(A,B)

	A	B	Result
Atom1	1	0	**0**
Atom2	1	1	**1**
Atom3	0	1	**0**

```
function Intersection(v::AbstractArray)
  return all(v)
end

@assert(Intersection ([false, false])==false)
@assert(Intersection ([true,  false])==false)
@assert(Intersection ([true,  true])==true)
@assert(Intersection ([false, true])==false)
```

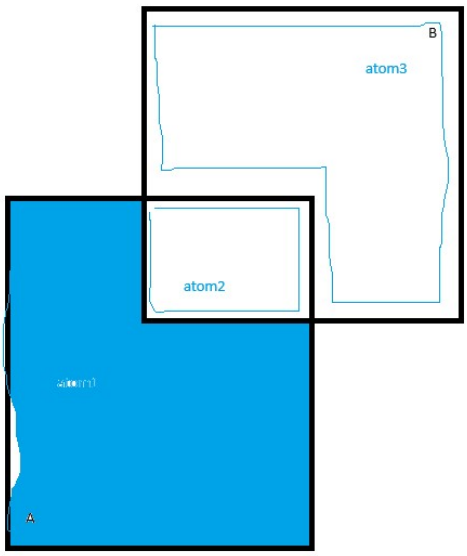


DIFFERENCE(A,B)

	A	B	Result
Atom1	1	0	**1**
Atom2	1	1	**0**
Atom3	0	1	**0**

```
function Difference(v::AbstractArray)
  return v[1] && !any(v[2:end])
end

@assert(Difference ([false,  false])==false)
@assert(Difference ([true,   false])==true)
@assert(Difference ([true,   true])==false)
@assert(Difference ([false,  true])==false)
```



XOR(A,B)

	A	B	Result
Atom1	1	0	**1**
Atom2	1	1	**0**
Atom3	0	1	**1**

```
function Xor(v::AbstractArray)
  return (length([it for it in v if it]) % 2)==1
end

@assert(Difference ([false, false])==false)
@assert(Difference ([true,  false])==true)
@assert(Difference ([true,  true])==false)
@assert(Difference ([false,  false])== true)
```