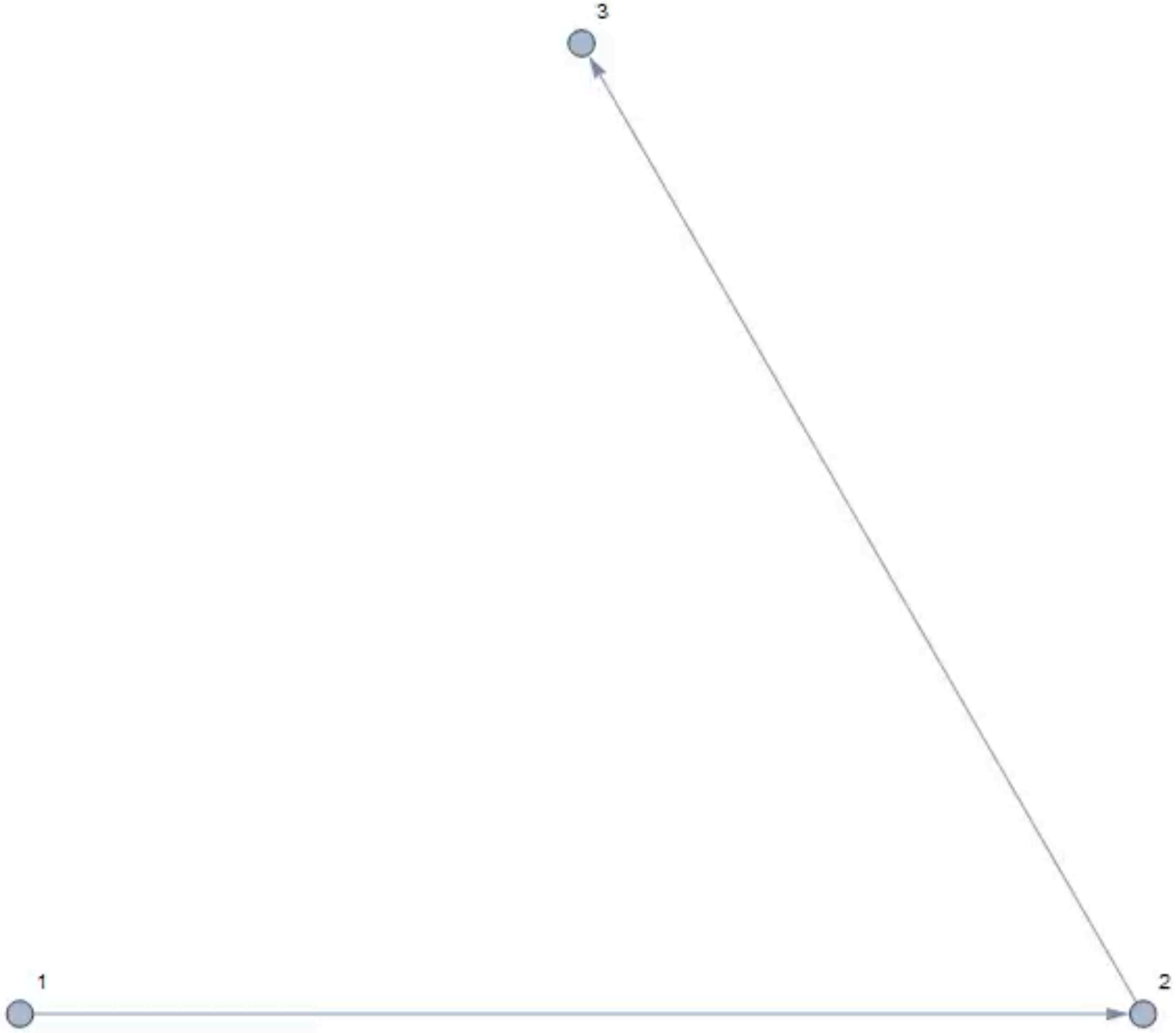
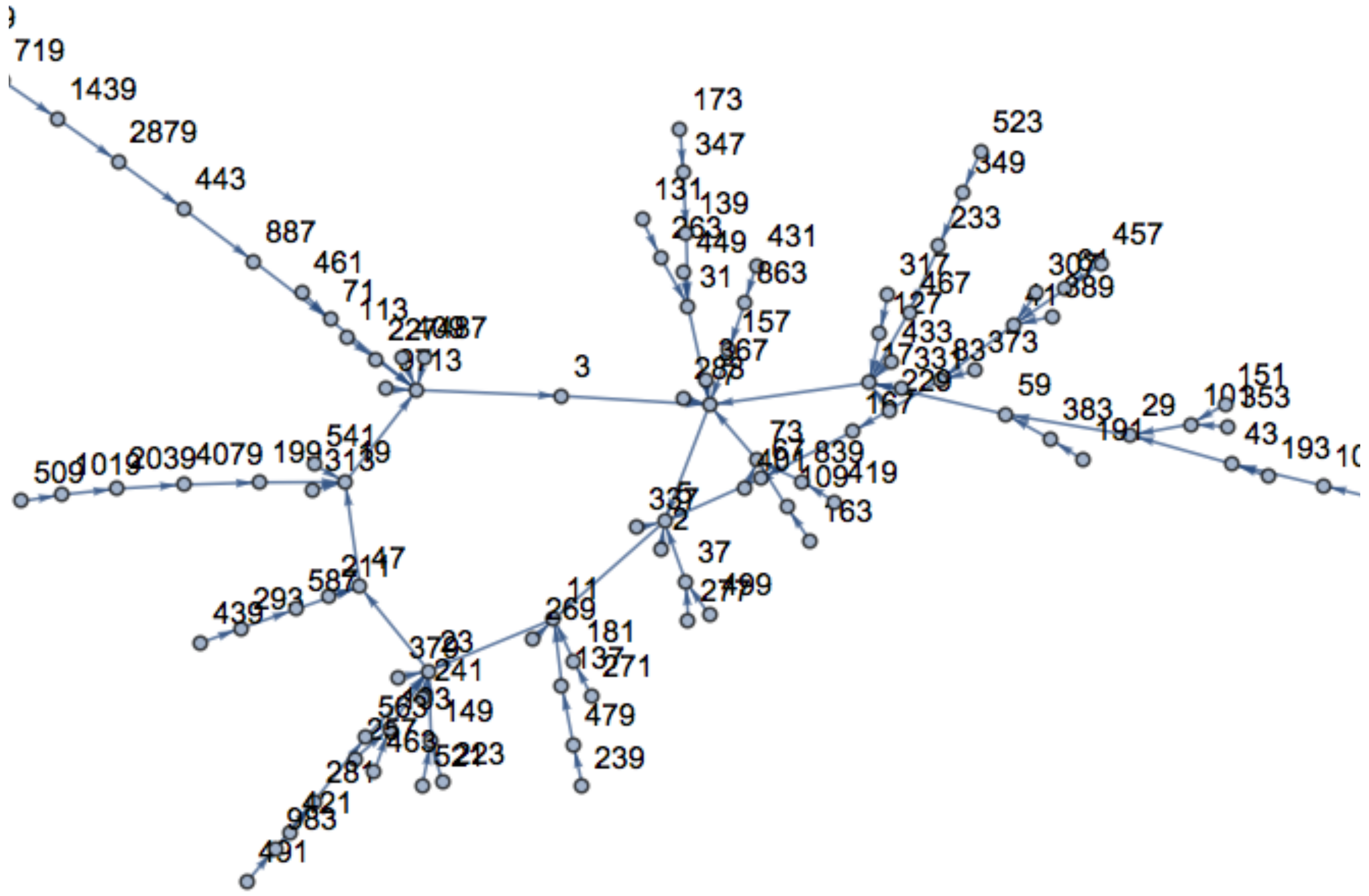
Extracting { structure }

Robbie VanDerzee, Friday July 30th





$$\sum_{n=1}^{k} 1 = \frac{k}{1}$$

$$\sum_{n=1}^{k} n = \frac{k}{1} \cdot \frac{k+1}{2}$$

$$\sum_{n=1}^{k} n^2 = \frac{k}{1} \cdot \frac{k+1}{2} \cdot \frac{2k+1}{3}$$

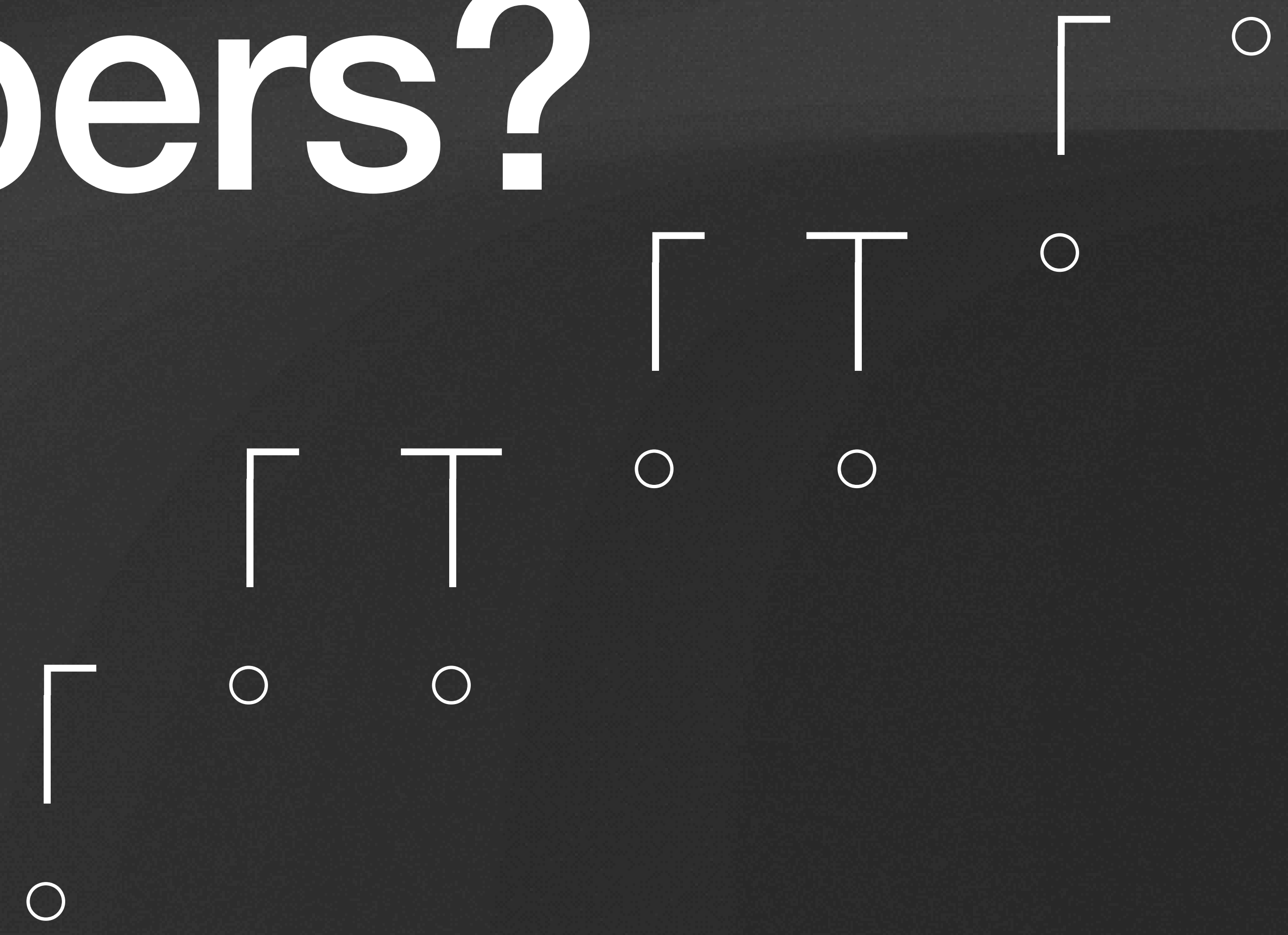
$$\sum_{n=1}^{k} n^3 ? = \frac{k}{1} \cdot \frac{k+1}{2} \cdot \frac{2k+1}{3} \cdot \frac{3k+1}{4}$$

What ways do we think about numbers?

$$v = \sum_{n=0}^{k} a_n b^n = a_0 b^0 + a_1 b^1 + \ldots + a_n b^n + \ldots + a_k b^k$$

$$b \geq 2, \quad 0 \leq a_n < b$$

What other ways could we think about numbers?



$$v = integer\langle k \rangle = \left\{ \; p_0, \, p_1, \, p_2, \, \ldots, \, p_k \; \right\}$$

$\mathbb{Z}^+(v)$	$\mathbb{Z}^+_{new}(v)$
0	$\varnothing$
1	$\{0\}$
2	$\{1\}$
3	$\{0,1\}$
4	$\{2\}$
45	$\{0,2,1\}$

$$v = \pm\ integer\langle k \rangle = \left\{ 0/1 \in p_{-1}, p_0, p_1, p_2, \ldots, p_k \right\}$$

$$v = \frac{p}{q} \ni (p, q) \in \mathbb{Z} \wedge gcd(p, q) \equiv 1$$

$$v = rational\langle k \rangle = \left\{ \pm\, p_{-1},\ \pm\ p_0,\ \pm\ p_1,\ \pm\ p_2, \ldots,\ \pm\ p_k \right\}$$

$\mathbb{R}^{\pm}(v)$	$\mathbb{R}^{\pm}_{new}(v)$
1/2	$\{0, -1\}$
1/3	$\{0,0, -1\}$
1/4	$\{0, -2,0\}$
1/5	$\{0,0,0, -1\}$
-5/7	$\{1,0,0,1, -1\}$
11/5	$\{0,0,0, -1,0,1\}$

$$v = complex\langle k \rangle = \left\{\ r_{-1},\ r_0,\ r_1,\ r_2,\ \ldots,\ r_k\ \right\}$$

$\mathbb{C}(v)$	$\mathbb{C}_{new}(v)$
$\sqrt{2}$	$\{\{0\}, \{0, -1\}\}$
$\sqrt{6}$	$\{\{0\}, \{0, -1\}, \{0,0, -1\}\}$
$\sqrt[3]{7.5}$	$\{\{0\}, \{1,0, -1\}, \{0,0, -1\} \{0,0, -1\}\}$
i	$\{\{1\}\}$
$\sqrt{-6}$	$\{\{1\}, \{0, -1\}, \{0,0, -1\}\}$
$\sqrt[\sqrt{2}]{2}$	$\{\{0\}, \{0,\{0, -1\}\}\}$

We can keep going, but there is an issue!

How can we decouple it?

How could we represent it better?

```
0 := ○

1  = (-1)^0 * (2)^0:
                ○
                ├── ○
                └── ○

2  = (-1)^0 * (2^(-1)^0) :=
                ○
                ├── ○
                └── ○
                    └── ○

30 = (-1)^0 * 2^(-1)^0 * 3^(-1)^0 * 5^(-1)^0 :=
                ○
                ├── ○
                ├── ○
                │   └── ○
                ├── ○
                │   └── ○
                └── ○
                    └── ○
```

How could we compute it?

```
class Atomic {
  public:

    Atomic() : atoms({}) {}

    Atomic(const std::initializer_list<Atomic> & Γ) : atoms {Γ} {}

    Atomic(const Atomic & ζ) = default;

    friend Atomic operator*(const Atomic & α, const Atomic & β) {
        if (!α.atoms.size()) return β;
        if (!β.atoms.size()) return α;
        if (!α.atoms.size() && !β.atoms.size()) return {};

        Atomic μ;
        μ.atoms.resize(std::max(α.atoms.size(), β.atoms.size()));

        for (int element = 0; element < μ.atoms.size(); element++) {
            μ.atoms[element] = (element < α.atoms.size() ? α.atoms[element] : Atomic {{}}) *
                               (element < β.atoms.size() ? β.atoms[element] : Atomic {{}});
        }
        return μ;
    }

    private:
        std::vector<Atomic> atoms;
    };
}
```

```
75 :=
○
├── ○
├── ○
├── ○
│   └── ○
└── ○
    ├── ○
    └── ○
        └── ○
```

```
3/2 :=
○
├── ○
├── ○
│   └── ○
│       └── ○
└── ○
    └── ○
```

```
17 :=
○
├── ○
├── ○
├── ○
├── ○
├── ○
├── ○
├── ○
└── ○
    └── ○
```

```
3/2 :=
○
├── ○
├── ○
│   └── ○
│       └── ○
└── ○
    └── ○
```

```
i^4 :=
○
└── ○
    ├── ○
    └── ○
        ├── ○
        └── ○
            └── ○
```

```
3i :=
○
├── ○
│   └── ○
├── ○
└── ○
    └── ○
```

```
3i_{2} :=
○
├── ○
│   └── ○
├── ○
└── ○
    └── ○
        └── ○
```

```
2*3 :=
○
├── ○
└── ○
    └── ○
```

```
*
○
├── ○
├── ○
└── ○
    └── ○
```

```
:=
○
├── ○
├── ○
│   └── ○
└── ○
    └── ○
```

How about leveraging temporary defeat?

```
friend Atomic operator+(const Atomic & α, const Atomic & β);
```

We’re further than we started!

Where else can we go?