Combinatorial Archenumbers

by Sven Nilsen, 2022

In this paper I present a method to talk about combinatorial similarity of natural numbers. The underlying theory was derived using Avatar Extensions, inspired by ongoing research on Path Semantical Quality and Seshatic vs Platonic bias in mathematical languages.

In combinatorics[1], one often uses natural numbers to describe the size of discrete spaces[2]. Discrete spaces vary significantly by structure and this makes it difficult to talk about general similarity. One of the problems is that a discrete structure can "factor out" another structure, hence have similarity e.g. in a visualization of the space, but numerically the difference in sizes of the spaces can be extremely large. Therefore, the size of a space tells little about combinatorial similarity.

However, is it possible to transform the size of a discrete space to some form where it is easier to talk about the combinatorial similarity? This is the idea behind Combinatorial Archenumbers.

Combinatorial Archenumbers focuses on products and powers, since it is computationally hard to cover sums. The principle is that one must compare "apples vs apples" and not "apples vs oranges". For example, when concatenating two discrete spaces into one, the size of the resulting discrete space is simply the sum of the sizes of the two separate discrete spaces:

sum_space[size] <=> add *For notation, see paper "Normal Paths"[3]*

When comparing spaces for structural similarity, it is required that sums are never used. Furthermore, only products and powers are assumed:

prod_space[size] <=> mul
pow_space[size] <=> pow

For example, discrete spaces representing pairs can be problematic, such as pairs that are constrained to a triangular connectivity matrix.

Once these assumptions hold, there is a simple algorithm `ty` that produces an archenumber:

```
ty : nat → nat
ty(n) = {
	if n < 2 {return n}
	n := n
	loop {
		m := trans(fact(n))
		if m == n {return m}
		n = m
	}
}

fact : nat → [nat]
```

Here, `fact` is ordered prime factorization and `trans` is defined on the next page.

The archenumbers produced by `ty` form the basis for talking about combinatorial similarity.

The definition of `trans`:

```
trans : [nat] → nat
trans(fact) = {
	if len(fact) == 0 {return 1}
	p := fact[0]
	res := 1
	n := 1
	for i [1, len(fact)) {
		if fact[i] == p {
			n += 1
		} else {
			res *= n * p
			p = fact[i]
			n = 1
		}
	}
	res *= n * p
	res
}
```

Combinatorial similarity is thought of as a set of 8 possible reductions over a base and an exponent:

```
nat × nat → nat
mul_ty(base, exp) = ty(base * exp)
mul_base_ty(base, exp) = ty(ty(base) * exp)
mul_exp_ty(base, exp) = ty(base * ty(exp))
mul_tty(base, exp) = ty(ty(base) * ty(exp))
pow_ty(base, exp) = ty(base ^ exp)
pow_base_ty(base, exp) = ty(ty(base) ^ exp)
pow_exp_ty(base, exp) = ty(base ^ ty(exp))
pow_tty(base, exp) = ty(ty(base) ^ ty(exp))
```

For example:

base exp: 8^2	base exp: 2^6
value: 64	value: 64
mul_ty: 6	mul_ty: 12
mul_base_ty: 12	mul_base_ty: 12
mul_exp_ty: 6	mul_exp_ty: 12
mul_tty: 12	mul_tty: 12
pow_ty: 12	pow_ty: 12
pow_base_ty: 12	pow_base_ty: 12
pow_exp_ty: 12	pow_exp_ty: 12
pow_tty: 12	pow_tty: 12

These reductions based on archenumbers share the greatest common divisor `2 * 3`.

The form `8^2` of `64` is thought of as a “lincombination” of `8`,
and the form `2^6` of `64` is thought of as “hexcombination” of `2`.

In an n-combination, the input can be interpreted geometrically as an infinitesimal n-gon[4].
The resulting combination makes the exponent the dominant geometry at global scale.
The archenumber is a restored balance of the exponent’s dominance over the base.

This restored balance in the example above, is represented as the product `2 * 3`.
It follows from the geometric interpretation that a circle is `1` and `0` is undefined.
The theory of Combinatorial Archenumbers was derived using Avatar Extensions[5].

References:

[1] "Combinatorics"
Wikipedia
https://en.wikipedia.org/wiki/Combinatorics

[2] "Discrete"
AdvancedResearch – Combinatorial phantom types for discrete mathematics
https://github.com/advancedresearch/discrete

[3] "Normal Paths"
Sven Nilsen, 2019
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/normal-paths.pdf

[4] "Regular polygon"
Wikipedia
https://en.wikipedia.org/wiki/Regular_polygon

[5] "Avatar Extensions"
AdvancedResearch – Summary page on Avatar Extensions
https://advancedresearch.github.io/avatar-extensions/summary.html