examples

July 1, 2021

0.1 Basics

First, import the package and everything from the enums module. `cgt` includes enums for every set of options so that your editor can provide you with the options available

[1]:
```
import cgt
from cgt.enums import *
```

We can define a `PositionParadigmFramework` under which we will model genomes. Choose to model oriented, circular genomes with five regions

[2]:
```
framework = cgt.PositionParadigmFramework(5, oriented=True, symmetry=SYMMETRY.
 ↪circular)
print(framework)
```

```
Framework for circular genomes with 5 oriented regions
```

Right away, we can look at the kinds of genomes we can represent

[3]:
```
genome = framework.random_genome()
print(genome)
```

```
1/10*(3,-4,-5)(4,5,-3) + 1/10*(1,2,3,-5,4)(5,-4,-1,-2,-3) +
1/10*(1,3,-1,-3)(2,4)(5,-5)(-4,-2) + 1/10*(1,4,3,-2,-5)(2,5,-1,-4,-3) +
1/10*(1,5,-2,-1,-5,2)(3,-3) + 1/10*(1,-5,-3,-2,4,-1,5,3,2,-4) +
1/10*(1,-4,5,2,-3,-1,4,-5,-2,3) + 1/10*(1,-3,-5,-1,3,5)(2,-2)(4,-4) +
1/10*(1,-2)(2,-1)(3,4,-3,-4) + 1/10*(1,-1)(2,-5,-4)(4,-2,5)
```

We can view the genome in other ways, too. Select a random permutation representing the above genome. These are referred to as genome instances

[4]:
```
instance = framework.random_instance(genome)
print(instance)
```

```
(1,5,-2,-1,-5,2)(3,-3)
```

Each genome has a canonical instance. Here canonical means the instance which maps region 1 to position 1. It is returned in one_row notation:

[5]:
```
canonical_instance = framework.canonical_instance(instance)
print(canonical_instance)
```

```
[1, 2, -4, 5, -3]
```

…but can be convered back to cycle notation easily.

[6]:
```
canonical_instance = framework.cycles(canonical_instance)
print(canonical_instance)
```

```
(3,-4,-5)(4,5,-3)
```

We can obtain the genome from a given instace, canonical or otherwise.

[7]:
```
new_genome = framework.genome(instance, format=cgt.FORMAT.formal_sum)
print(genome == new_genome)
```

```
True
```