Engine

James Long

March 7, 2009

First we load up the required modules. We will be testing the main engine module.

```
(load "lib/engine")
```

1 Selection Procedures

1.1 selection-rws

Implements roulette wheel selection. Given a number N, where N is between 0 and the sum of all fitnesses, linearly search the genomes and find the genome occupying the specified slot, where slot lengths are the genome's fitness.

```
(define-test selection-rws
  (let ((pop (make-population 3)))
    (genotype-fitness-set! (car pop) 50)
    (genotype-fitness-set! (cadr pop) 25)
    (genotype-fitness-set! (caddr pop) 5)
    (assert-equal (selection-rws pop 25) (car pop))
    (assert-equal (selection-rws pop 50) (cadr pop))
    (assert-equal (selection-rws pop 70) (cadr pop))
    (assert-equal (selection-rws pop 76) (caddr pop))))
```

1.2 selection-sus

Implement stochastic universal selection. This build on top of roulette wheel selection: instead of passing a random value to **??**, select mutiple genomes at once, dividing the fitness space evenly and selecting the genomes at those points. Typically, you would want to select a whole new population at once, which means the spread of your fitness scores controls the selection behaviour (higher spread, more variance, and vice versa).

```
(define-test selection-sus
  (let ((pop (make-population 3)))
    (genotype-fitness-set! (car pop) 50)
```

```
(genotype-fitness-set! (cadr pop) 25)
(genotype-fitness-set! (caddr pop) 5)
(let ((vec-pop (list->vector (selection-sus pop 16))))
  (assert-equal (vector-ref vec-pop 0) (car pop))
  (assert-equal (vector-ref vec-pop 10) (cadr pop))
  (assert-equal (vector-ref vec-pop 15) (caddr pop))))
```