`github.com/sjsyrek/presentations/lambda-calculus`

1. Vist the code repo at the above link
2. Download or clone the repo
3. Open `lambda.html` and a browser console
4. Alternatively, you can copy and paste the code from `javascript/lambda.js` directly into the console
5. Confirm that all tests have passed

λ calculus*

*for those who can't be bothered to learn it

Steven Syrek

`@sjsyrek`

definition

```
<expression>  := <name> | <function> | <application>
<function>    := λ <name>. <expression>
<application> := <expression> <expression>
```

- `x`
- `λx. x`
- `λx. λy. x y`
- `(λx. x) y`
- `(λx. x)(λy. y)`

operations

α-substitution

β-reduction

η-conversion

- `(λa. a) ≡ (λz. z) ≡ (λPoop. Poop) ≡ (λ💩.💩)`
- `(λx. λy. x y) p q → (λy. p y) q → p q`
- `(λx. x x)(λx. x x) → (λx. x x)(λx. x x) → …`
- `(λx . f x) ↔ f`

identity combinator

- `id ≡ (λx. x)`
- `id = \x -> x`
- `id = x => x`

boolean combinators

- `true ≡ (λx. λy. x)`
- `false ≡ (λx. λy. y)`
- `and ≡ (λa. λb. a b false)`
- `or  ≡ (λa. λb. a true b)`
- `not ≡ (λa. a false true)`

reducing expressions

- `and true false ≡`
- `(λa. λb. a b (λx. λy. y))(λx. λy. x)(λx. λy. y) ≡`
- `(λb. (λx. λy. x) b (λx. λy. y))(λx. λy. y) ≡`
- `(λx. λy. x)(λx. λy. y)(λx. λy. y) ≡`
- `(λy. (λx. λy. y))(λx. λy. y) ≡`
- `(λx. λy. y) ≡`
- `false`

predicates

- `isZero ≡ (λx. x false not false)`
- `isZero(n)(true)(false)`
- `if ≡ λp. λx. λy. p x y`
- `if(isZero(n))(‹then›)(‹else›)`
- `if ≡ id`
- *$P\ E_1\ E_2$ = if P is true then E_1 else E_2*

numbers

- `0 ≡ (λf. λx. x)`
- `1 ≡ (λf. λx. f(x))`
- `2 ≡ (λf. λx. f(f(x)))`
- `3 ≡ (λf. λx. f(f(f(x))))`
- `4 ≡ (λf. λx. f(f(f(f(x)))))`
- `5 ≡ (λf. λx. f(f(f(f(f(x))))))`
- …

enumeration

- succ ≡ (λn. λf. λx. f(n f x))
- pred ≡ (λn. n (λp. λz. z(succ (p true))(p true)) (λz. z 0 0) false)
- succ 1 ≡ (λn. λf. λx. f(n f x))(λf. x. f(x)) ≡ (λf. λx. f(f(x))) ≡ 2
- pred 1 ≡ (λn. n (λp. λz. z(succ (p true))(p true)) (λz. z 0 0) false)(λf. λx. f(x)) ≡ (λf. λx. x) ≡ 0

arithmetic

- add ≡ (λn. λm. λf. λx. n f(m f x))
- add ≡ (λn. λm. m succ n)
- sub ≡ (λm. λn. n pred m)
- mult ≡ (λn. λm. λf. n(m f))
- mult ≡ (λn. λm. m (add n) 0)
- exp ≡ (λx. λy. y x)

arithmetic

- add 1 2 ≡ (λn. λm. λf. λx. n f(m f x))(λf. λx. f(x))(λf. λx. f(f(x))) = (λf. λx. f(f(f(x)))) ≡ 3
- sub 2 1 ≡ (λm. λn. n (λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) m)(λf. λx. f(f(x)))(λf. λx. f(x)) = (λf. λx. f(x)) ≡ 1
- mult 3 0 ≡ (λn. λm. λf. n(m f))(λf. λx. f(f(f(x))))(λf. λx. x) = (λf. λx. x) ≡ 0
- exp 2 3 ≡ (λx. λy. y x)(λf. λx. f(f(x)))(λf. λx. f(f(f(x)))) = (λf. λx. f(f(f(f(f(f(f(f(x))))))))) ≡ 8

recursion

- `fix ≡ (λy. (λx. y(x x))(λx. y(x x)))`
- `F ≡ (λf. λn. ((isZero n) 1 (mult n (f(pred n)))))`
- `fact ≡ (fix F)`
- `((λy. (λx. y(x x))(λx. y(x x))) F)`
- `(λx. F(x x))(λx. F(x x))`
- `(F((λx. F(x x))(λx. F(x x))))`
- `(F(fix F))`

factorial

- `fact 2`
- `(fix F) 2`
- `((λy. (λx. y(x x))(λx. y(x x))) F) 2`
- `((λx. F(x x))(λx. F(x x))) 2`
- `(F((λx. F(x x))(λx. F(x x)))) 2`
- `(F(fix F)) 2`
- `((λf. λn. ((isZero n) 1 (mult n (f(pred n)))))(fix F)) 2`
- `(λn. ((isZero n) 1 (mult n ((fix F)(pred n))))) 2`
- `((isZero 2) 1 (mult 2 ((fix F)(pred 2))))`
- `(mult 2 ((fix F)(pred 2)))`
- `(mult 2 ((fix F) 1))`
- `(mult 2 1)`
- `2`

factorial (expanded)

- ((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f))n(f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))(λf. λx. f(f(x)))
- ((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y))n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))(λf. λx. f(f(x)))
- ((λx. (λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p(λx. λy. x))) (λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))(x x))(λx. (λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))(x x)))(λf. λx. f(f(x)))
- ((λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y))n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y))n)))))((λx. (λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y))n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))(x x))(λx. (λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))(x x))))(λf. λx. f(f(x)))
- ((λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y))n)))))((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))))(λf. λx. f(f(x)))
- ((λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n) (λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))))(λf. λx. f(f(x)))
- (λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n)))))))((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))) (λf. λx. f(f(x)))
- (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y))(λf. λx. f(f(x))))(λf. λx. f(x))((λn. λm. λf. n(m f))(λf. λx. f(f(x)))(((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y) (λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y))(λf. λx. f(f(x)))))))
- ((λn. λm. λf. n(m f))(λf. λx. f(f(x)))(((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y))(λf. λx. f(f(x))))))
- ((λn. λm. λf. n(m f))(λf. λx. f(f(x)))(((λy. (λx. y(x x))(λx. y(x x)))(λf. λn. (((λx. x (λx. λy. y)(λa. a (λx. λy. y)(λx. λy. x))(λx. λy. y)) n)(λf. λx. f(x))((λn. λm. λf. n(m f)) n (f((λn. n (λp. λz. z((λn. λf. λx. f(n f x))(p (λx. λy. x)))(p (λx. λy. x)))(λz. z (λf. λx. x)(λf. λx. x))(λx. λy. y)) n))))))(λf. λx. f(x))))
- ((λn. λm. λf. n(m f))(λf. λx. f(f(x)))(λf. λx. f(x)))
- (λf. λx. f(f(x)))

github.com/sjsyrek/malc

@sjsyrek