Recursion and Induction

Recursive definition of factorial

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot (n-1)! & \text{if } n > 0. \end{cases}$$

Squaring a number recursively without multiplication.

Squaring a number recursively without multiplication.

Compute n^2 given $(n-1)^2$

Squaring a number recursively without multiplication.

Compute n^2 given $(n-1)^2$

$(n-1)^2 =$

Squaring a number recursively without multiplication.

Compute n^2 given $(n-1)^2$

$$(n-1)^2 = n^2 - 2n + 1$$

Squaring a number recursively without multiplication.

Compute n^2 given $(n-1)^2$

$$(n-1)^2 = n^2 - 2n + 1$$

$$n^2 = (n-1)^2 + 2n - 1$$

Squaring a number recursively without multiplication.

Compute n^2 given $(n-1)^2$

$$(n-1)^2 = n^2 - 2n + 1$$

$$n^2 = (n-1)^2 + 2n - 1$$

Program this in Scheme

Prove `square` is correct.

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Prove `square` is correct.

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

The correctness proof of a recursive function is by mathematical induction.

Mathematical induction:

Mathematical induction:

- Base case corresponds to base case in code.

Mathematical induction:

- Base case corresponds to base case in code.
- Inductive case corresponds to recursive call in code.

Mathematical induction:

- Base case corresponds to base case in code.
- Inductive case corresponds to recursive call in code.
- Use *code inspection* to convert from Scheme to traditional infix notation in both cases.

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Base case

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Base case

Code inspection: `(square 0)` returns 0.

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Base case

Code inspection: `(square 0)` returns 0.

Math: $0^2 = 0$

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Base case

Code inspection: `(square 0)` returns 0.

Math: $0^2 = 0$

Therefore, correct in base case.

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Prove that

`(square n)` terminates with value n^2

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Prove that

`(square n)` terminates with value n^2

assuming that

`(square (- n 1))` terminates with value $(n-1)^2$

as the inductive hypothesis.

Inductive case

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ $\langle$Code inspection$\rangle$

`(square (- n 1))` $+(n+n)-1$

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

$(n-1)^2+(n+n)-1$

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

$(n-1)^2+(n+n)-1$

$=$ ⟨Math⟩

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

$(n-1)^2+(n+n)-1$

$=$ ⟨Math⟩

$n^2-2n+1+2n-1$

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

$(n-1)^2+(n+n)-1$

$=$ ⟨Math⟩

$n^2-2n+1+2n-1$

$=$ ⟨Math⟩

```
(define square
  (lambda (n)
    (if (= n 0)
        0
        (+ (square (- n 1))
           (- (+ n n) 1)))))
```

Inductive case

Value returned by `(square n)`

$=$ ⟨Code inspection⟩

`(square (- n 1))` $+(n+n)-1$

$=$ ⟨Inductive hypothesis⟩

$(n-1)^2+(n+n)-1$

$=$ ⟨Math⟩

$n^2-2n+1+2n-1$

$=$ ⟨Math⟩

n^2 ∎

The `cond` function

```
(cond ((condition1) expr1)
      ((condition2) expr2)
      ((condition3) expr3)
      (else expr4))
```

The `sum-of-first` function

The `sum-of-first` function

```
> (sum-of-first 4)
10
> (sum-of-first 5)
15
```

The `sum-of-first` function

```
> (sum-of-first 4)
10
> (sum-of-first 5)
15
```

`(sum-of-first n)` returns $1 + 2 + \cdots + n$

The `num-digits` function

The `num-digits` function

```
> (num-digits 59274)
5
> (num-digits 81)
2
```

The `num-digits` function

```
> (num-digits 59274)
5
> (num-digits 81)
2
```

`(num-digits n)` returns the number of digits in n