Naturais

Paradigma de Programação Funcional

Marco A L Barbosa

Conteúdo

O estudo utilizando apenas este material **não é suficiente** para o entendimento do conteúdo. Recomendamos a leitura das referências no final deste material e a resolução (por parte do aluno) de todos os exercícios indicados.

Introdução

Introdução

- Um número natural é atômico ou composto?

Introdução

- Um número natural é atômico ou composto?
  - Atômico quando usado em operações aritméticas
  - Composto quando uma iteração precisa ser feita baseado no valor do número

Introdução

- Um número natural é atômico ou composto?
  - Atômico quando usado em operações aritméticas
  - Composto quando uma iteração precisa ser feita baseado no valor do número
- Se um número natural pode ser visto como dado composto
  - Quais são as partes que compõe o número?
  - Como (de)compor um número?

Definição

Definição

- Um número **Natural** é
  - 0; ou
  - `(add1 n)` onde *n* é um número **Natural**

Definição

- Um número **Natural** é
  - 0; ou
  - `(add1 n)` onde *n* é um número **Natural**
- Baseado nesta definição, criamos um template para funções com números naturais

```
(define (fun-for-natural n)
  (cond
    [(zero? n) ...]
    [else ...
          n
          (fun-for-natural (sub1 n))]))
```

Definição

```
;; as funções add1, sub1 e zero? são pré-definidas

;; compõe um novo natural a partir de um existente
;; semelhante ao cons
> (add1 8)
9
;; decompõe um natural
;; semelhante ao rest
> (sub1 8)
7
;; verifica se um natural é 0
;; semelhante ao empty?
> (zero? 8)
#f
> (zero? 0)
#t
```

Exemplos

Exemplo 4.1

Dado um número natural n, defina uma função que some os números naturais menores ou iguais a n.

Passo 1: Contrato, propósito e cabeçalho

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n
(define (soma n) 0)
```

Passo 1: Contrato, propósito e cabeçalho

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n
(define (soma n) 0)
```

Passo 2: Exemplos

```
(check-equal? (soma 0) 0)
(check-equal? (soma 1) 1) ; (+ 1 0)
(check-equal? (soma 3) 6) ; (+ 3 (+ 2 (+ 1 0)))
```

Passo 1: Contrato, propósito e cabeçalho

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n
(define (soma n) 0)
```

Passo 2: Exemplos

```
(check-equal? (soma 0) 0)
(check-equal? (soma 1) 1) ; (+ 1 0)
(check-equal? (soma 3) 6) ; (+ 3 (+ 2 (+ 1 0)))
```

Passo 3: Template

```
(define (soma n)
  (cond
    [(zero? n) ...]
    [else ... n (soma (sub1 n))]))
```

Passo 4: Corpo (baseado nos exemplos, completamos o template)

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n

(check-equal (soma 0) 0)
(check-equal (soma 1) 1) ; (+ 1 0)
(check-equal (soma 3) 6) ; (+ 3 (+ 2 (+ 1 0)))

(define (soma n)
  (cond
    [(zero? n) ...]
    [else ... n (soma (sub1 n))]))
```

Passo 4: Corpo (baseado nos exemplos, completamos o template)

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n

(check-equal (soma 0) 0)
(check-equal (soma 1) 1) ; (+ 1 0)
(check-equal (soma 3) 6) ; (+ 3 (+ 2 (+ 1 0)))

(define (soma n)
  (cond
    [(zero? n) ...]
    [else ... n (soma (sub1 n))]))

(define (soma n)
  (cond
    [(zero? n) 0]
    [else ... n (soma (sub1 n))]))
```

Passo 4: Corpo (baseado nos exemplos, completamos o template)

```
;; Natural -> Natural
;; Soma todos os números naturais de 0 até n

(check-equal (soma 0) 0)
(check-equal (soma 1) 1) ; (+ 1 0)
(check-equal (soma 3) 6) ; (+ 3 (+ 2 (+ 1 0)))

(define (soma n)
  (cond
    [(zero? n) ...]
    [else ... n (soma (sub1 n))]))

(define (soma n)
  (cond
    [(zero? n) 0]
    [else ... n (soma (sub1 n))]))

(define (soma n)
  (cond
    [(zero? n) 0]
    [else (+ n (soma (sub1 n)))]))
```

Exemplo 4.2

Dado um número natural n, defina uma função que devolva a lista `(list n n-1 n-2 ...  1)`.

Definição

- As vezes queremos utilizar um caso base diferente de 0
- Podemos generalizar a definição de número natural para incluir um limite inferior diferente de 0

Definição

- Um número **Inteiro>=a** é
  - *a*; ou
  - `(add1 n)` onde *n* é um número **Inteiro>=a**

Definição

- Um número **Inteiro>=a** é
  - *a*; ou
  - `(add1 n)` onde *n* é um número **Inteiro>=a**

- Template

```
(define (fun-for-inteiro>=a n)
  (cond
    [(<= n a) ...]
    [else ...
          n
          (fun-for-inteiro>=a (sub1 n))]))
```

Exemplo 4.3

[htdp 11.4.7] Escreva uma função `tem-divisor-entre-2-e-i?`, que receba como parâmetros dois números naturais, *n* e *i*. Se *n* não é divisível por nenhum número entre 2 (inclusive) e *i* (inclusive), a função deve devolver verdadeiro, caso contrário falso. Utilizando a função `tem-divisor-entre-2-e-i?`, defina uma função `primo?`, que verifica se um número natural é primo. Um número natural é primo se ele tem exatamente dois divisores distintos: 1 e ele mesmo.

Referências

Referências

- Vídeos Naturals