Combinação de templates

Paradigma de Programação Funcional

Marco A L Barbosa

Este trabalho está licenciado com uma Licença Creative Commons - Atribuição-CompartilhaIgual 4.0 Internacional.

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

- Qual template utilizar quando a função consome dois ou mais tipos de dados?

Introdução

- Qual template utilizar quando a função consome dois ou mais tipos de dados?
  - Se apenas um dado é definido por mais que uma cláusula (como por exemplo, uma lista), utilizamos o template correspondente
  - Se mais que dois dados de entrada são definidos por mais que uma cláusula, devemos fazer uma combinação dos templates

Exemplos

Exemplo 5.1

Dados duas listas `lsta` e `lstb`, defina uma função que verifique se `lsta` é prefixo de `lstb`, isto é `lstb` começa com `ltsa`.

Exemplo 5.1

- Passo 1: Assinatura, propósito e cabeçalho

```
;; Lista Lista -> Boolean
;; Devolve true se ltsa é prefixo de lstb. false caso contrário.
(define (prefixo? ltsa ltsb) false)
```

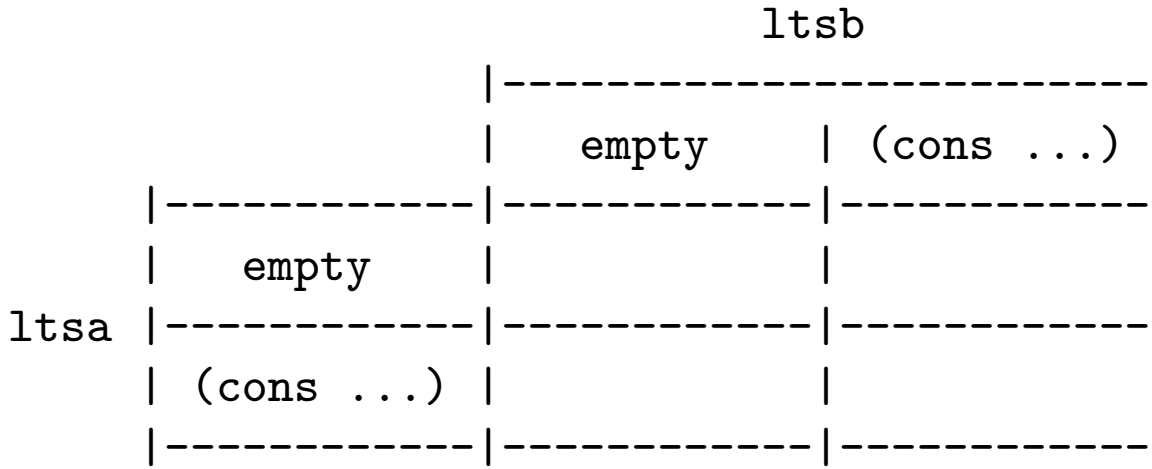
Exemplo 5.1

- Passo 2: Exemplos
  - Temos ter pelo menos um exemplo para cada combinação das definições dos dados de entrada
  - `ltsa` pode ser `empty` ou um `cons`
  - `ltsb` pode ser `empty` ou um `cons`
  - Como garantir que não vamos esquecer nenhum caso?

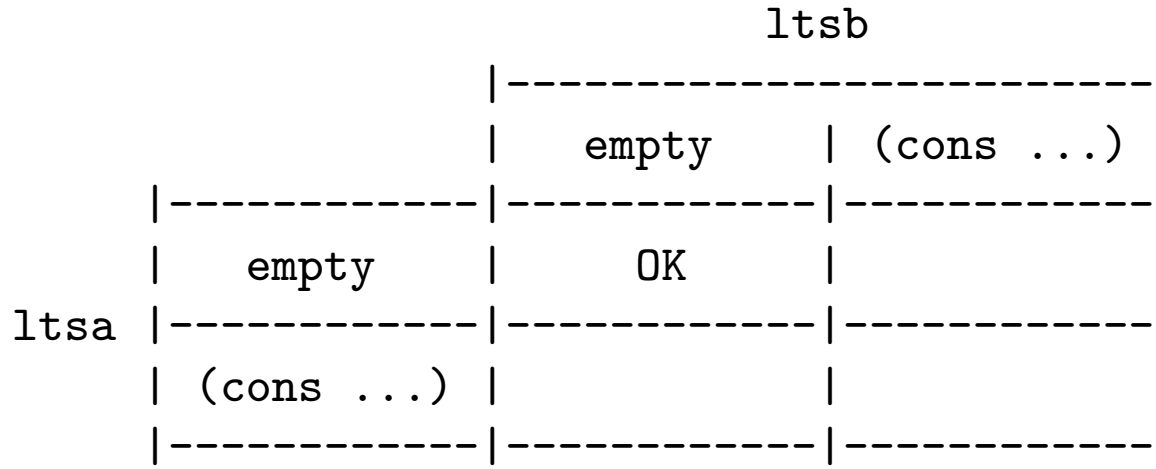
Exemplo 5.1

- Passo 2: Exemplos
  - Temos ter pelo menos um exemplo para cada combinação das definições dos dados de entrada
  - `ltsa` pode ser `empty` ou um `cons`
  - `ltsb` pode ser `empty` ou um `cons`
  - Como garantir que não vamos esquecer nenhum caso? Fazendo uma tabela

Exemplo 5.1

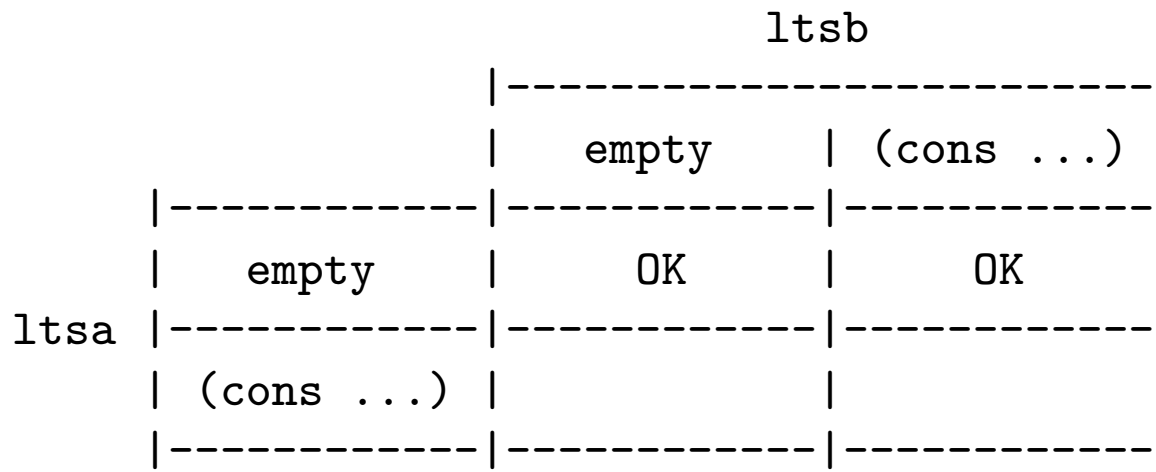


Exemplo 5.1



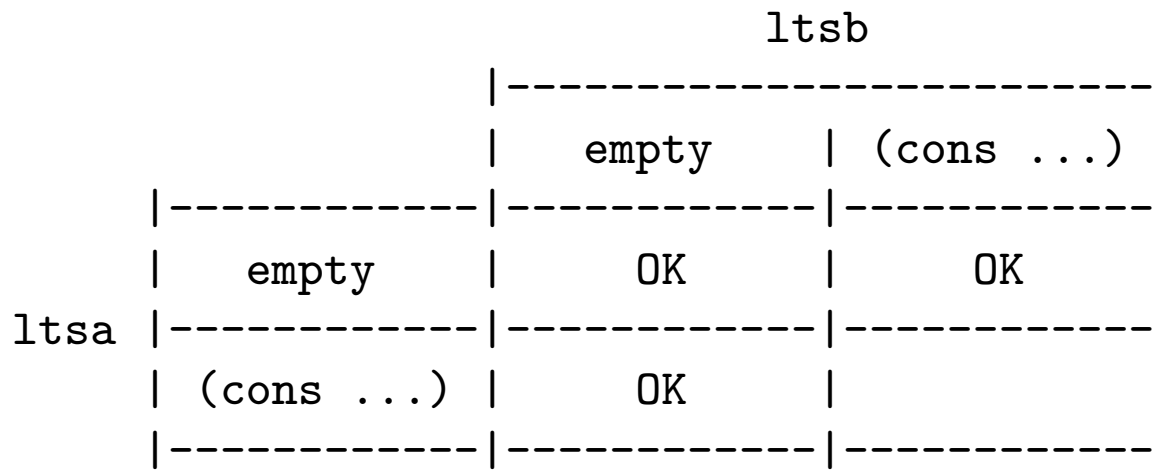

```
(check-equal? (prefixo? empty empty) true)
```

Exemplo 5.1




```
(check-equal? (prefixo? empty empty) true)
(check-equal? (prefixo? empty (list 3 2 1)) true)
```

Exemplo 5.1




```
(check-equal? (prefixo? empty empty) true)
(check-equal? (prefixo? empty (list 3 2 1)) true)
(check-equal? (prefixo? (list 3 2 1) empty) false)
```

Exemplo 5.1

ltsa \ ltsb	empty	(cons ...)
empty	OK	OK
(cons ...)	OK	OK

```
(check-equal? (prefixo? empty empty) true)
(check-equal? (prefixo? empty (list 3 2 1)) true)
(check-equal? (prefixo? (list 3 2 1) empty) false)
(check-equal? (prefixo? (list 3 4) (list 3 4)) true)
(check-equal? (prefixo? (list 3 4) (list 3 5)) false)
(check-equal? (prefixo? (list 3 4) (list 3 4 6 8)) true)
(check-equal? (prefixo? (list 3 5) (list 3 4 6 8)) false)
(check-equal? (prefixo? (list 3 4 5) (list 3 4)) false)))
```

Exemplo 5.1

- Passo 3: template
  - Baseado na tabela, vamos criar um template

Exemplo 5.1

- Passo 3: template
  - Baseado na tabela, vamos criar um template

```
(define (prefixo? lsta ltsb)
  (cond
    [(and (empty? lsta) (empty? ltsb)) ...]
    [(and (empty? lsta) (cons? ltsb)) ... lstb ...]
    [(and (cons? lsta) (empty? ltsb)) ... lsba ...]
    [else ... lsta ... ltsb ...]))
```

Exemplo 5.1

- Passo 3: template
  - Baseado na tabela, vamos criar um template

```
(define (prefixo? lsta ltsb)
  (cond
    [(and (empty? lsta) (empty? ltsb)) ...]
    [(and (empty? lsta) (cons? ltsb)) ... lstb ...]
    [(and (cons? lsta) (empty? ltsb)) ... lsba ...]
    [else ... lsta ... ltsb ...]))
```

  - Este template é muito complicado...

Exemplo 5.1

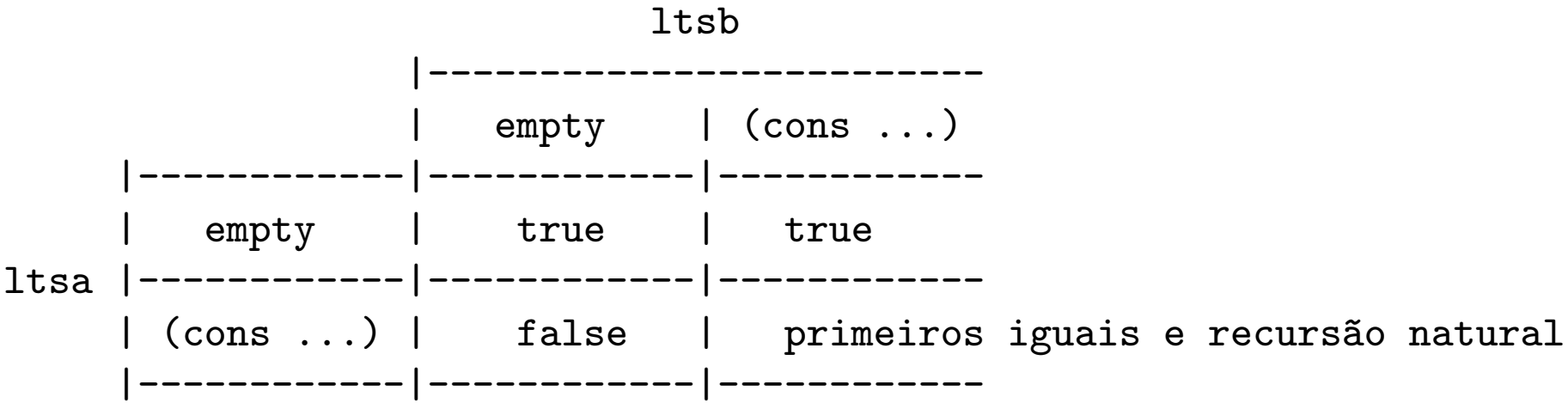
- Passo 3: template
  - Baseado na tabela, vamos criar um template

```
(define (prefixo? lsta ltsb)
  (cond
    [(and (empty? lsta) (empty? ltsb)) ...]
    [(and (empty? lsta) (cons? ltsb)) ... lstb ...]
    [(and (cons? lsta) (empty? ltsb)) ... lsba ...]
    [else ... lsta ... ltsb ...]))
```

  - Este template é muito complicado...
  - Baseado nos exemplos, vamos preencher a tabela e derivar um template mais simples

Exemplo 5.1

```
(check-equal? (prefixo? empty empty) true)
(check-equal? (prefixo? empty (list 3 2 1)) true)
(check-equal? (prefixo? (list 3 2 1) empty) false)
(check-equal? (prefixo? (list 3 4) (list 3 4)) true)
(check-equal? (prefixo? (list 3 4) (list 3 5)) false)
(check-equal? (prefixo? (list 3 4) (list 3 4 6 8)) true)
(check-equal? (prefixo? (list 3 5) (list 3 4 6 8)) false)
(check-equal? (prefixo? (list 3 4 5) (list 3 4)) false)))
```



Exemplo 5.1

- Simplificando . . .

ltsa \ ltsb	empty	(cons ...)
empty	true	
(cons ...)	false	primeiros iguais e recursão natural

- Template (observe que alguma parte do corpo já foi escrita)

```
(define (prefixo? lsta ltsb)
  (cond
    [(empty? ltsa) true]     ;; os casos foram
    [(empty? lstb) false]    ;; escolhidos por ordem
    [else ...                ;; de simplicidade
          (first lsta)
          (first ltsb)
          (prefixo? (rest ltsa) (rest ltsb))]))
```

Exemplo 5.1

- Passo 4: Corpo

```
(define (prefixo? lsta ltsb)
  (cond
    [(empty? ltsa) true]
    [(empty? lstb) false]
    [else (and
            (equal? (first lsta) (first ltsb))
            (prefixo? (rest ltsa) (rest ltsb)))]))
```

Exemplo 5.2

Defina uma função que encontre o k-ésimo elemento de uma lista.

Referências

Referências

- Vídeos 2 one-of