Fundamentos de Algoritmos

CIC/UFRGS

2006/1

Mais em definições auto-referenciáveis

- foi visto como definir árvores de ascendência, definindo dois campos do tipo nó que indicam os dois pais

- árvores de descendência requerem uma forma de identificar um número arbitário de “filhos”

Roteiro

- Listas de estruturas, listas em estruturas
- Definindo funções para definições mutuamente referenciáveis
- Mais sobre páginas web

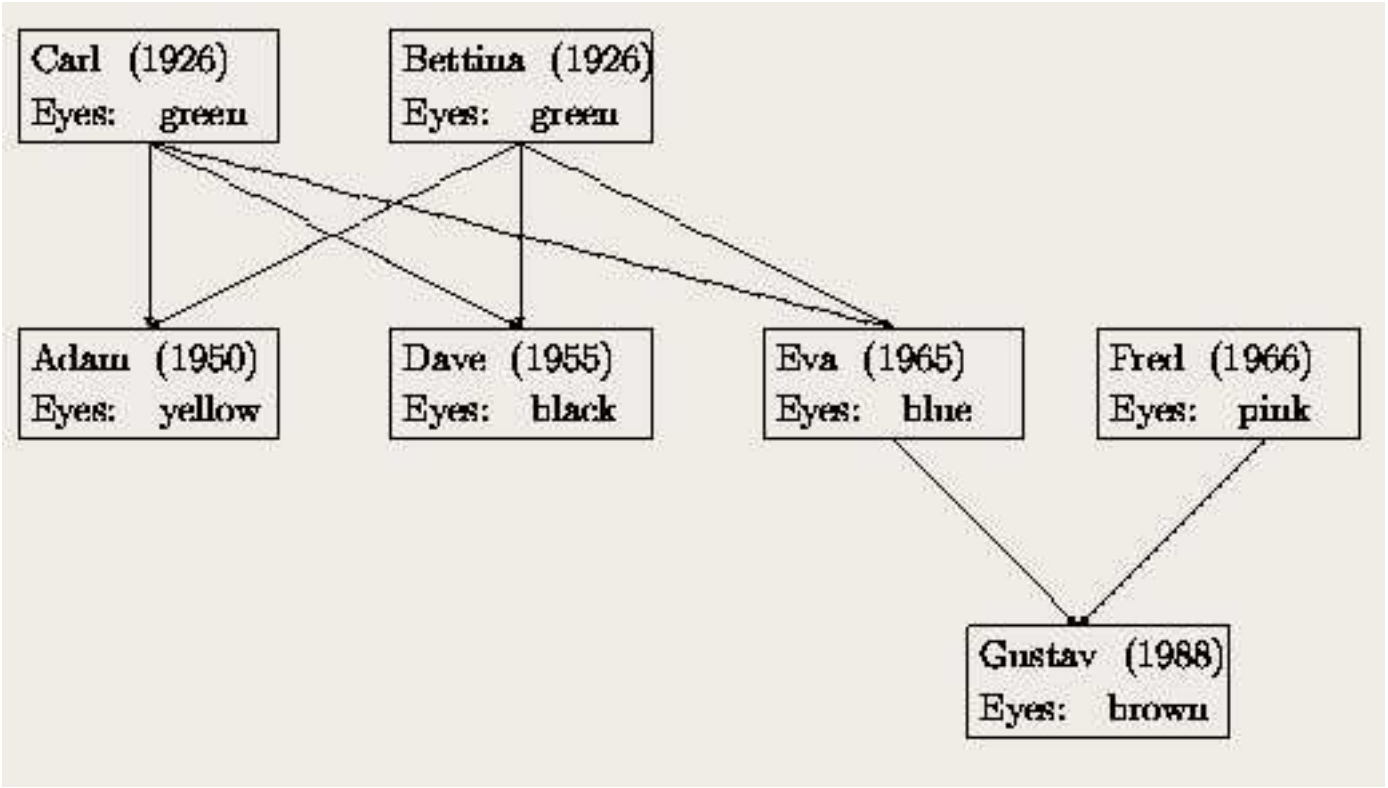
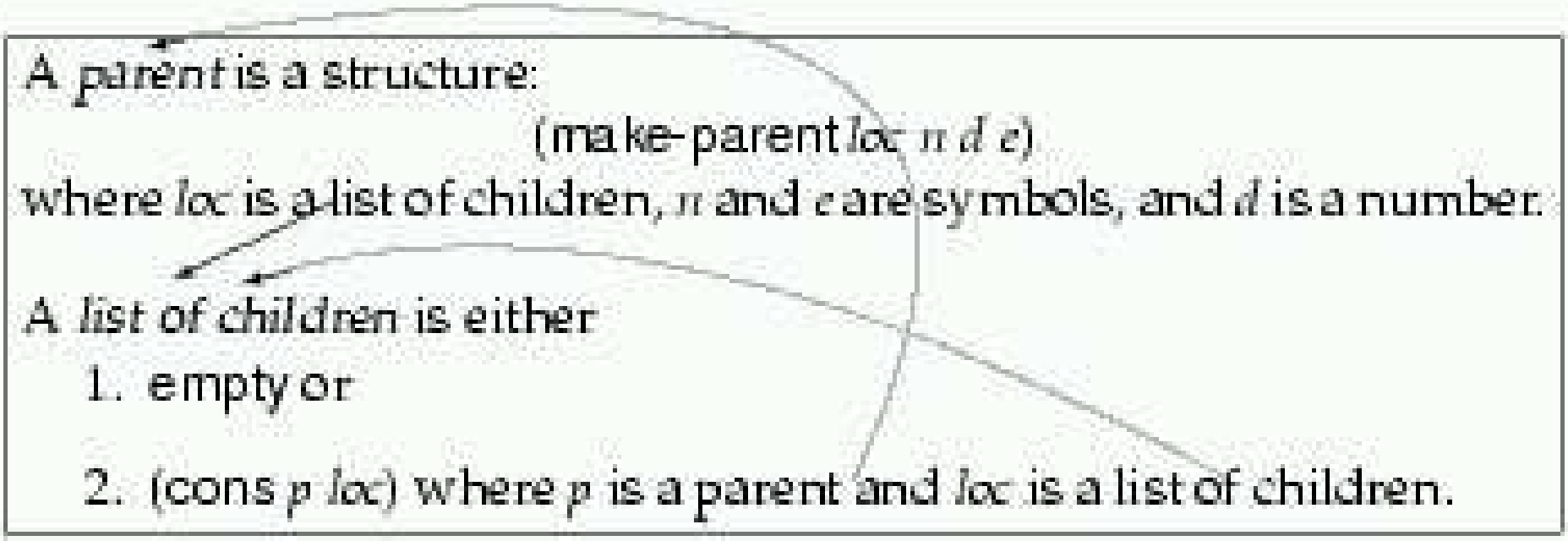


Figura 1: Árvore de hereditariedade: tipo descendente

Estrutura de cada indivíduo

- (**define-struct** parent (filhos nome data olhos))
- um parent pode contar um número arbitário de filhos. Então o campo `filhos` deve conter um número indeterminado de nós.
- Um parent é uma estrutura: (make-parent lf n d o) onde
  lf é uma estrutura do tipo lista de filhos
  n e o são símbolos e d é número.
- lista de filhos é:
  - empty ou
  - (cons p lf) onde p é um parent e lf é uma lista de filhos

Definições Mutuamente Referenciáveis



Árvore de descendência

```
;; Geração nova:
(define Gustav (make-parent empty 'Gustav 1988 'brown))
(define Fred&Eva (list Gustav))

;; Geração intermediária:
(define Adam (make-parent empty 'Adam 1950 'yellow))
(define Dave (make-parent empty 'Dave 1955 'black))
(define Eva (make-parent Fred&Eva 'Eva 1965 'blue))
(define Fred (make-parent Fred&Eva 'Fred 1966 'pink))
(define Carl&Bettina (list Adam Dave Eva)

;; Geração antiga:
(define Carl (make-parent Carl&Bettina 'Carl 1926 'green))
(define Bettina (make-parent Carl&Bettina 'Bettina 1926 'green))
```

Determinar descendente de olhos azuis

```
;; descendente-olhos-azuis? parent -> boolean
;; determina se p1 ou algum de seus descendentes
;; possui olhos azuis
(define (descendente-olhos-azuis? p1)...)
```

- (boolean=? (descendente-olhos-azuis? Gustav) false)
- (boolean=? (descendente-olhos-azuis? Eva) true)
- (boolean=? (descendente-olhos-azuis? Bettina) true)

Template para funções que determina descendente de olhos azuis

```
;; descendente-olhos-azuis? parent -> boolean
(define (descendente-olhos-azuis? p1)...)
  (cond
    [(symbol=? (parent-olhos p1) 'blue) true]
    [else
     ... (parent-filhos p1) ...
     ... (parent-nome p1) ...
     ... (parent-data p1) ...
     ... (parent-olhos p1) ...
    ]
  )
)
```

Descendente com olhos azuis

```
;; descendente-olhos-azuis? parent -> boolean
;; determina se p1 ou algum de seus descendentes
;; possui olhos azuis
(define (descendente-olhos-azuis? p1)
  (cond
    [(symbol=? (parent-olhos p1) 'blue) true]
    [else (filho-olhos-azuis? (parent-filhos p1))]))
```

Verifica filho com olhos azuis

```
(define (filho-olhos-azuis? lf)
  (cond
    [(empty? lf) ...]
    [else
      ... (first lf) ...
      ... (filho-olhos-azuis? (rest lf)) ...]))
```

descendente de olhos azuis: versão com cond

```
;; descendente-olhos-azuis? parent -> boolean
;; determina se p1 ou algum de seus descendentes possui olhos azuis
(define (descendent-olhos-azuis? p1)
  (cond
    [(symbol=? (parent-olhos p1) 'blue) true]
    [else (filho-olhos-azuis? (parent-filhos p1))]))

;; filho-olhos-azuis? : lista-de-filhos  ->  booleano
;; determina se um filho da lista, ou  possui olhos azuis
(define (filho-olhos-azuis? lf)
  (cond
    [(empty? lf) false]
    [else (cond
        [(descendent-olhos-azuis? (first lf)) true]
        [else (filho-olhos-azuis? (rest lf))])]))
```

descendente olhos azuis: versão or

```
;; descendente-olhos-azuis? parent -> boolean
;; determina se p1 ou algum de seus descendentes possui olhos azuis
(define (descendent-olhos-azuis? p1)
  (or (symbol=? (parent-olhos p1) 'blue)
      (filho-olhos-azuis? (parent-filhos p1))))

;; filho-olhos-azuis? : lista-de-filhos  ->  booleano
;; determina se um filho da lista possui olhos azuis
(define (blue-eyed-children? lf)
  (cond
    [(empty? lf) false]
    [else (or (descendente-olhos-azuis? (first lf))
              (filho-olhos-azuis? (rest lf)))]))
```

Exercícios

- 15.1.2 Desenvolva a função `quanto-longe`. A função determina o quanto longe está o mais próximo descendente de olhos azuis. Se nenhum é encontrado, retorne `false`.

- 15.1.3 Desenvolva a função `conte-descendentes`, o qual conta o número de descendentes. Desenvolva uma variante que conta o no atual como descendente e uma variante que não.

- 15.1.4 Desenvolva a função `cor-dos-olhos` o qual produz uma lista da cor dos olhos dos descendentes. Uma cor pode ocorrer mais que uma vez na lista.

Projetando Funções para Definições Mutuamente Referenciáveis

Fase: Análise de dados e projeto

Objetivo: formular um grupo de definições de dados

Atividade: desenvolva um grupo de definições de dados mutuamente recursivas; identifique explicitamente todas as referências entre as definições de dados

Fase: Template

Objetivo: determine esquemas para um grupo de funções

Atividade: desenvolva tantos templates quanto as definições de dados simultâneas

Fase: Corpo da função

Objetivo: determine um grupo de funções condicional

Atividade: formule uma expressão para cada cláusula cond; use funções auxiliares quando necessário

Mais sobre páginas web

(**define-struct** wp (header body))

Uma página web é uma estrutura do tipo:
(make-wp h p)
onde h é um símbolo e p é um documento web.

Um documento web é:

1. empty

2. (cons s p) onde s é um símbolo e p é um documento

3. (cons w p) onde w é uma página web e p é um documento.

Exercícios

- **Exercício 15.3.1** Defina a função *size*, que consome um página Web e retorna o número de símbolos (words) que esta contém.

- **Exercícios 15.3.2.** Defina a função *wp-to-file*. A função consome uma página Web e produz uma lista de símbolos. A lista contém todas as palavras no corpo e todos os cabeçalhos das páginas associadas no mesmo domínio. Os corpo do texto das páginas associadas é ignorado.

- **Exercício 15.3.3** Defina a função *occurs*. Ele consome um símbolo e uma página Web e determina se o símbolo ocorrem em algum lugar da página, incluindo páginas associadas no mesmo domínio.

Exercícios

- **Exercícios 15.3.4.** Defina a função *find*. A função consome uma página Web e um símbolo. Ela produz *false*, se o símbolo não ocorre no corpo da página ou nas associadas. Se o símbolo ocorre pelo menos uma vez, é produzida uma lista dos cabeçalhos que foram encontrados no caminho ao símbolo.

  Dica: Defina uma auxiliar *find* que produz apenas *true* quando uma página Web contém a palavra desejada. Use esta para definir *find*. Alternativamente, use $boolean?$ para determinar se uma recursão natural de find produziu uma lista ou um booleano. Então calcule o resultado novamente. Iremos discutiresta segunda técnica chamada *backtracking*, em seções adiante.