

Fundamentos de lenguajes de programación

La relación entre Inducción y Programación

EISC. Facultad de Ingeniería. Universidad del Valle

Mayo de 2019

Contenido

- 1 Especificación Recursiva de datos
 - Especificación inductiva
 - Especificación mediante gramáticas en forma BNF
- 2 Especificación recursiva de programas
- 3 Los conceptos de Alcance y ligadura de una variable

Contenido

- 1 Especificación Recursiva de datos
 - Especificación inductiva
 - Especificación mediante gramáticas en forma BNF
- 2 Especificación recursiva de programas
- 3 Los conceptos de Alcance y ligadura de una variable

Especificación Recursiva de datos

- Cuando se escribe un procedimiento, se debe definir que clase de valores se espera como entrada y como salida.
- Ejemplo, la función suma tiene como entrada dos números naturales y tiene como salida un número natural.
- Los datos en las funciones recursivas, pueden tener también definiciones recursivas que faciliten la programación.

Especificación Recursiva de datos

Técnicas

Existe dos técnicas para la definición recursiva de datos:

- 1 Especificación inductiva
- 2 Especificación mediante gramáticas.

Especificación inductiva

Definición

Se define un conjunto S , el cual es el conjunto más pequeño que satisface las siguientes dos propiedades:

- 1 Algunos valores específicos que deben estar en S .
- 2 Si algunos valores están en S , entonces otros valores también están en S .

Especificación inductiva

Números pares

- 1 Si $n = 2$ entonces n es par
- 2 Si n es par, entonces $n + 2$ también es par.

Lista de números

- 1 *empty* es una lista de números
- 2 Si n es un número y l es una lista entonces $(n \ l)$ es una lista de números

Fix

$(n \ l)$

Especificación inductiva

Especificación formal

Ahora formalmente:

Números pares

1 $2 \in S$ $\leftarrow B_{base}$

2
$$\frac{n \in S}{(n+2) \in S}$$

Lista de números

1 $() \in S$ $\leftarrow B_{base}$

2
$$\frac{l \in S, n \in \mathbb{N}}{(n \ l) \in S}$$

Especificación inductiva

Ejemplo

Demuestre que $(1(2(3())))$ es una lista de números.

Solución

1 $1 \in \mathbb{N}, (2(3())) \in S$

2 $(1(2(3()))) \in S$

Se puede seguir hasta llegar al caso fundamental $() \in S$

$$(1 (2 (3 ())))$$

$$\begin{array}{l} 1) \quad (2 (3 ())) \in S \wedge 1 \in \mathbb{N} \\ \hline (1 (2 (3 ()))) \in S \end{array}$$

$$\begin{array}{l} 2) \quad (3 ())) \in S \wedge 2 \in \mathbb{N} \\ \hline (2 (3 ())) \in S \end{array}$$

$$\begin{array}{l} 3) \quad () \in S \wedge 3 \in \mathbb{N} \\ \hline (3 ())) \in S \end{array}$$

Especificación inductiva

Especificación formal

Ahora realicemos la especificación inductiva de:

1 Una lista de número pares

2 Múltiplos de 5

$\leftarrow \{() \in S, x \in \mathbb{N} \dots y \in P$
 $\vdash 5 \in S$

Especificación inductiva

Lista de números pares

1 $() \in S$

2 $\frac{l \in S, n \in \mathbb{N}}{(2n+1) \in S}$

\rightarrow

$$\begin{array}{l} c) () \in S \\ \frac{l \in S, n \in \mathbb{N}}{(2n+1) \in S} \end{array}$$

$$\begin{array}{l} d) 2 \in P \\ b) \frac{x \in P}{x+2 \in P} \end{array}$$

Múltiplos de 5

1 $5 \in S$

2 $\frac{n \in S}{(n+5) \in S}$

Especificación inductiva

Ejercicios

Indique que tipo de conjuntos están definidos por las siguiente reglas:

1 $(0, 1) \in S, \frac{(n,k) \in S}{(n+1,k+7) \in S}$

$(1, 8), (2, 15), (3, 22)$

2 $(0, 1) \in S, \frac{(n,k) \in S}{(n+1, 2k) \in S}$

$(0, 1), (1, 2), (2, 2^2), (x, 7x+1)$
 $(3, 2^3) \rightarrow (x, 2^x)$

→ 3 $(0, 1, 0) \in S, \frac{(i,j,k) \in S}{(i+1, j+2, i+j) \in S}$

Para cada una de las especificaciones dé dos ejemplos numéricos que las cumplan.

$(0, 1, 1)$
 $(0, 1, 0) \rightarrow (1, 3, 4) \rightarrow (2, 5, 7) \rightarrow (3, 7, 10) \rightarrow (4, 9, 13)$
 $(5, 11, 16)$
 $(x, 2x+1, 3x+1)$

Especificación mediante gramáticas

- Una forma sencilla de especificar datos recursivos es con gramáticas regulares en forma Backus-Nour.
- Las gramáticas se componen de:
 - 1 Símbolos no terminales, que son aquellos que se componen de otros símbolos, son conocidos como categorías sintácticas
 - 2 Símbolos terminales: corresponden a elementos del alfabeto
 - 3 Reglas de producción

Especificación mediante gramáticas

- Alfabeto: Conjunto de símbolos, ejemplo $\Sigma = \{a, b, c, \dots\}$
- Reglas de producción: Construcción del lenguaje:
 - Cerradura de Kleene: $\{a\}^* = \{\epsilon, \{a\}, \{a, a\}, \{a, a, a\} \dots\}$
 - Cerradura positiva: $\{b\}^+ = \{\{b\}, \{b, b\}, \{b, b, b\} \dots\}$

$$a^* \Rightarrow \epsilon, a, aa, aqa, \dots$$

$$b^+ \Rightarrow b, bb, bbb, \dots$$

Especificación mediante gramáticas

Lista números

- a) $\langle \text{lista-de-enteros} \rangle ::= ()$
 $::= (\langle \text{int} \rangle \langle \text{lista-de-enteros} \rangle)$
 ↑ ↑
- b) $\langle \text{lista-de-enteros} \rangle ::= () | (\langle \text{int} \rangle \langle \text{lista-de-enteros} \rangle)$
- c) $\langle \text{lista-de-enteros} \rangle ::= (\langle \text{int} \rangle)^*$

Especificación mediante gramáticas

Árbol Binario

```
<arbol-binario> ::= <int>  
                  ::= (<simbolo> <arbol-binario> <arbol-binario>  
                      >)
```

Ejemplos de árboles binarios son:

1

(foo 1 2)

(foo 1 (bar 2 3) 3)

`<arbol-binario> ::= <int>
::= (<simbolo> <arbol-binario>
 <arbol-binario>)`

`'(foo (foo 1 2) 3)`
 $\underbrace{\hspace{1.5cm}}$ $\underbrace{\hspace{1.5cm}}$
 cadr caddr

`(define suma-arbol
 (lambda (arb)
 (cond
 [(number? arb) arb]
 [else
 +
 (suma-arbol (cadr arb))
 (suma-arbol (caddr arb))
]
))))`

Dr Rocket
lists

Especificación mediante gramáticas

Expresión calculo λ

$\langle \text{lambda-exp} \rangle ::= \langle \text{identificador} \rangle$
 $::= (\text{lambda } (\langle \text{identificador} \rangle) \langle \text{lambda-exp} \rangle)$
 $::= (\langle \text{lambda-exp} \rangle \langle \text{lambda-exp} \rangle)$
 $\langle \text{identificador} \rangle ::= \langle \text{letra} \rangle^+$

Ejemplo de cálculo λ :

```
'(lambda (x) (x y))  
'((lambda (y) (z y)) 1 )  
'(x y) —  
'x
```

Contenido

- 1 Especificación Recursiva de datos
 - Especificación inductiva
 - Especificación mediante gramáticas en forma BNF
- 2 Especificación recursiva de programas
- 3 Los conceptos de Alcance y ligadura de una variable

Especificación recursiva de programas

- La definición inductiva o mediante gramáticas de los conjuntos de datos sirve de guía para desarrollar procedimientos que operan sobre dichos datos
- La estructura de los programas debe seguir la estructura de los datos
- Para esto realizamos especificación recursiva de programas, la idea es utilizar el principio del subproblema más pequeño.

Especificación recursiva de programas

Ejemplo

Una función estándar de Dr Racket es `list-length`, la cual nos retorna el tamaño de una lista. Para el diseño de esta función debemos retornar a la especificación recursiva de las listas

`Lista ::= () | (numero Lista)`

Se debe considerar entonces el caso base de la lista vacía, en el cual retornamos 0.

Especificación recursiva de programas

Ejemplo

De acuerdo a esto el diseño de list-length es:

```
;;list-length: lista -> numero
(define list-length
  (lambda (lst)
    (if (null? lst)
        0
        ...
    )
  )
)
```

Tomando en cuenta la segunda parte de la definición, se debe sumar 1 al tamaño si no encontramos la lista vacía.

Especificación recursiva de programas

Ejemplo

Por lo tanto la función list-length es:

```
;;list-length: lista -> numero
(define list-length
  (lambda (lst)
    (if (null? lst)
        0
        (+ 1 (list-length (cdr lst)))))
)
```

De acuerdo a la especificación recursiva de listas ¿Que se puede decir de este diseño?

Especificación Recursiva de programas

Un árbol binario

Recordando la definición de los árboles vista anteriormente:

```
<arbol-binario> ::= <int>  
                ::= (<simbolo> <arbol-binario>  
                    <arbol-binario>)
```

Este árbol será representado así:

```
(define arbolA '(k (h 5 3) (t (s 10 11) 12)))  
(define arbolB 2)
```

En Dr Racket el operador `'(...)` va generar una lista, toda palabra será convertida en símbolo y todo `(..)` será convertido en lista.

Especificación Recursiva de programas

Un árbol binario

Procedimiento sum-arbol:

```
(define sum-arbol
  (lambda (arbol)
    (if (number? arbol)
        arbol
        (+ (sum-arbol (cadr arbol))
           (sum-arbol (caddr arbol))
          )
        )
    )
  )
```

Especificación Recursiva de programas

Lista números

Procedimiento lista de números:

```
;;BNF
;;<lista-numeros> ::= '() | <int> <lista-numeros>
(define suma-lista-numeros
  (lambda (l)
    (if (number? (car l))
        (car l)
        (+
         (suma-lista-numeros (cadr l))
         (suma-lista-numeros (caddr l))
        )
    )
  )
)
```

Especificación recursiva de programas

Resumen

En el diseño de programas para datos recursivos tenga en cuenta:

- 1 Identifique el caso terminal (base) o donde termina la especificación recursiva
- 2 La idea es pasar de estados no terminales a uno terminal
- 3 La estrategia es llamar recursivamente la función desde un estado no terminal para llegar a uno terminal

Especificación recursiva de programas

Ejercicio

Tomando en cuenta la especificación recursiva de listas, diseñe funciones:

- 1 nth-element. (nth-element '(1 2 3) 2) retorna 2.
- 2 remove-first (remove-first '(1 2 3)) retorna '(2 3)

La especificación mediante gramáticas de listas de números es:

`Lista ::= () | (numero Lista)`

Contenido

- 1 Especificación Recursiva de datos
 - Especificación inductiva
 - Especificación mediante gramáticas en forma BNF
- 2 Especificación recursiva de programas
- 3 Los conceptos de Alcance y ligadura de una variable

Los conceptos de Alcance y Ligadura de una variable

- El concepto de variable es fundamental en los lenguajes de programación
- Una variable puede ser declarada y posteriormente ligada o referenciada

- Declaración:

```
(lambda (x) ...)  
(let ((x ...)) ...)
```

- Referencia:

```
(f x y)
```

Los conceptos de Alcance y Ligadura de una variable

- Una variable esta ligada al lugar donde se declara
- El valor ligado o referenciado por la variable es su denotación
- Cada lenguaje de programación tiene asociadas unas reglas de ligadura que determinan a qué declaración hace referencia cada variable
- Dependiendo del momento de aplicación de las reglas antes o durante la ejecución, los lenguajes se denominan de alcance estático o alcance dinámico

Los conceptos de Alcance y Ligadura de una variable

- Si la expresión e es una variable, la variable x ocurre libre iff x es igual a e .
- Si la expresión e es de la forma $(\lambda(y) e')$ entonces la variable x ocurre libre iff si y es distinto que x y x ocurre libre en e' .
- Si la expresión e es de la forma (e_1, e_2) , entonces x ocurre libre iff si está ocurre libre en e_1 o e_2

Los conceptos de Alcance y Ligadura de una variable

Dada la definición anterior, indique en las siguiente expresiones si x ocurre libre o no.

- $'(\text{lambda } (x) (\text{lambda } (y) x))$
- $'((\text{lambda } (z) (\text{lambda } (y) x) x))$
- $'((\text{lambda } (y) (\text{lambda } (y) x)) ((\text{lambda } (z) x) x))$

Los conceptos de Alcance y Ligadura de una variable

Para examinar si x ocurre libre o no en una expresión, por ejemplo:

```
'(lambda (x) (lambda (y) z))
```

Para el diseño debemos considerar la gramática del calculo λ

```
<lambda-exp> ::= <identificador>  
::= (lambda (<identificador>) <lambda-exp>)  
::= (<lambda-exp> <lambda-exp>)
```

```
<identificador> ::= <letra>+
```

Los conceptos de Alcance y Ligadura de una variable

Determinar si una variable ocurre libre

De esta forma se diseña un procedimiento que evalúa si **var** ocurre libre en **exp**.

```
(define occurs-free?  
  (lambda (var exp)  
    (cond  
      ((symbol? exp) (eqv? var exp))  
      ((eqv? (car exp) 'lambda)  
       (and (not (eqv? (caadr exp) var))  
            (occurs-free? var (caddr exp))))  
      (else (or (occurs-free? var (car exp))  
                (occurs-free? var (cadr exp)))))))
```

Los conceptos de Alcance y Ligadura de una variable

Se define como el alcance de una variable como la región dentro del programa en el cual ocurren todas las referencias a dicha variable.

```
(define x                               ; Variable x1
  (lambda (x)                           ; Variable x2
    (map
      (lambda (x)                       ; Variable x3
        (+ x 1))                       ; Ref x3
      x)))                             ; Ref x2
(x '(1 2 3))                           ; Ref x1
```

Los conceptos de Alcance y Ligadura de una variable

Ejemplo:

```
(lambda (z)
  ((lambda (a b c)
    (a (lambda (a)
      (+ a c))
      b))
    (lambda (f x)
      (f (z x))))))
```

```
(lambda (z)
  ((lambda (a b c)
    (a (lambda (a)
      (+ a c))
      b))
    (lambda (f x)
      (f (z x)) ) ) )
```

Los conceptos de Alcance y Ligadura de una variable

Cual es el valor de la siguiente expresión:

```
(let ((x 3) (y 4))  
  (+ (let ((x (+ y 5)))  
      (* x y))  
    x)  
)
```

Los conceptos de Alcance y Ligadura de una variable

Cual es el valor de la siguiente expresión:

```
(let ((x 6)(y 7))
  (*
    (let ((y 8))
      (+
        (let ((x 6) (y x))
          (+ x
            (let ((y 3) (x y)) (+ x (+ 2 y)))
          )
        )
      y)
    )
  (let ((x 4)) (- y x))
)
```


Los conceptos de Alcance y Ligadura de una variable

Cual es el valor de la siguiente expresión:

```
(let ((x 6)
      (y 7))
  (+ (let ((x (- y 6)))
      (* x y))
     x)
)
```

Próxima sesión

Abstracción de datos (Capítulo 2 EOPL)