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.



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.



Números pares

- I 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





Especificación formal

Ahora formalmente:

Números pares

- $1 2 \in S \leftarrow B_9 s_e$
- $\frac{n \in S}{(n+2) \in S}$

Lista de números

- 1 () $\in S$ \leftarrow Byse 2 $\frac{I \in S, n \in N}{(n \mid 1) \in S}$



Ejemplo

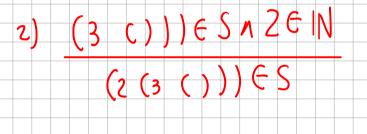
Demuestre que (1(2(3()))) es una lista de números.

Solución

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

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





3) $(3 c)) \in S$

Especificación formal

Ahora realicemos la especificación inductiva de:

- Una lista de número pares ← () ∈ S, × ∈ N ··· y ∈ P
- 2 Múltiplos de 5 \leftarrow 5 \in 5



Lista de números pares

- **1** () ∈ *S*
 - $\frac{1 \in S, n \in N}{(2n \ 1) \in S}$



Múltiplos de 5

- **1** 5 ∈ *S*
- $\frac{n \in S}{(n+5) \in S}$



Ejercicios

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

Para cada una de las especificaciones dé dos ejemplos numéricos

- 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
 - Reglas de producción



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





Lista números



Árbol Binario

Ejemplos de árboles binarios son:

```
1
(foo 1 2)
(foo 1 (bar 2 3) 3)
```



```
<arbol-binario> ::= <int>
::= (<simbolo> <arbol-binario>
    <arbol-binario
       (define suma-arbol
            (lambda (arb)
              ( cond
                [(number? arb) arb)]
                [else
                      (suma-arbol (cadr arb))
                      (suma-arbol (caddr arb))
```

```
Expresión calculo \lambda

<lambda-exp>\colon=:= <identificador>
::= (lambda (<identificador>) <lambda-exp>)
::= (<lambda-exp> <lambda-exp>)

<identificador> ::= <letra>+

Ejemplo de cálculo \lambda:
```

```
'(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



- 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.



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.



Ejemplo

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

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



Ejemplo

Por lo tanto la función list-length es:

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



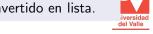
Un árbol binario

Recordando la definición de los árboles vista anteriormente:

Este árbol será representado así:

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

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



Un árbol binario

Procedimiento sum-arbol:



Lista números

Procedimiento lista de números:

Resumen

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

- 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



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



- 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 \times y)
```



- 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



- 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



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

- '(lambda (x) (lambda (y) x)
- '((lambda (z) (lambda (y) x) x)
- \bullet '((lambda (y) (lambda (y) x)) ((lambda (z) x) x))



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>+
```



Determinar si una variable ocurre libre

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



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



Ejemplo:

```
(lambda (z)

((lambda (a b c)

(a (lambda (a)

(+ a c))

b))

(lambda (f x)

(f (z x)))))))
```



Cual es el valor de la siguiente expresión:

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



Cual es el valor de la siguiente expresión:

```
(1et ((x 6)(y 7))
   (let ((y 8))
      (let ((x 6) (y x))
           x (let ((y 3) (x y)) (+ x (+ 2 y)))
```

Cual es el valor de la siguiente expresión:

Próxima sesión

Abstracción de datos (Capitulo 2 EOPL)

