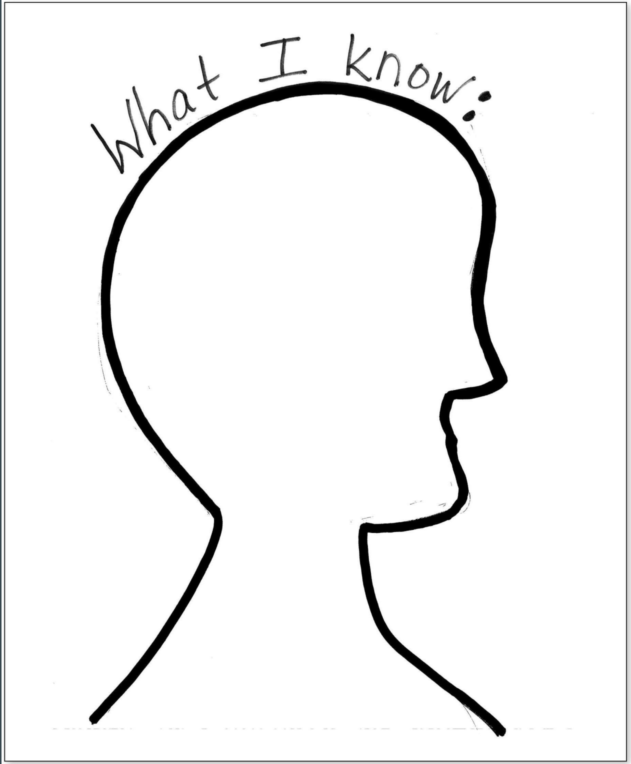
Universidad Nacional Autónoma de México
Facultad de Ciencias
Lenguajes de Programación

•••

Karla Ramírez Pulido

Inferencia de tipos


What I know:

Notación

Símbolo: $[\![\cdot]\!]$

El tipo de la expresión entre “dobles corchetes”

Para hacer inferencia de tipos en lenguajes con tipificado implícito.

Restricciones de tipo

EXPRESIÓN	RESTRICCIONES GENERADAS
n, cuando n es un numeral	[[n]] = number
true	[[true]] = boolean
false	[[false]] = boolean
(add1 e)	[[(add1 e)]] = number y [[e]] = number
(+ e1 e2)	[[(+ e1 e2)]] = number y [[e1]] = [[e2]] = number

Restricciones de tipo

EXPRESIÓN	**RESTRICCIONES GENERADAS**
(zero? e)	[[(zero? e)]] = boolean y [[e]] = number
(ncons e1 e2)	[[(ncons e1 e2)]] = list(num) donde [[e1]] = number y [[e2]] = list(num)
(nfirst e)	[[(nfirst e)]] = number y [[e]] = list(num)
(nrest e)	[[(nrest e)]] = list(num) y [[e]] = list(num)
(nempty? e)	[[(nempty? e)]] = boolean y [[e]] = list(num)

Restricciones de tipo

EXPRESIÓN	RESTRICCIONES GENERADAS
nempty	[[nempty]] = list(num)
(if c t e)	[[(if c t e)]] = [[t]] ó [[(if c t e)]] = [[e]] y [[c]] = boolean
(lambda (x) b)	[[(lambda(x) b)]] = [[x]] → [[b]]
(f a)	[[f]] = [[a]] → [[(f a)]]

Restricciones:

Expression at Node	Generated Constraints
n, where n is a numeral	$[\![n]\!] =$ `number`
true	$[\![$true$]\!]=$`boolean`
false	$[\![$false$]\!]=$`boolean`
$(add1\ e)$	$[\![(add1\ e)]\!]=$`number` $[\![e]\!]=$`number`
$(+\ e1\ e2)$	$[\![(+\ e1\ e2)]\!]=$`number` $[\![e1]\!] =$ `number` $[\![e2]\!] =$ `number`
$(zero?\ e)$	$[\![(zero?\ e)]\!]=$`boolean` $[\![e]\!] =$ `number`
$(ncons\ e1\ e2)$	$[\![(ncons\ e1\ e2)]\!]=$`list(num)` $[\![e1]\!] =$ `number` $[\![e2]\!] =$ `list(num)`
$(nfirst\ e)$	$[\![(nfirst\ e)]\!]=$`number` $[\![e]\!] =$ `list(num)`
$(nrest\ e)$	$[\![(nrest\ e)]\!]=$`list(num)` $[\![e]\!] =$ `list(num)`
$(nempty?\ e)$	$[\![(nempty?\ e)]\!]=$`boolean` $[\![e]\!] =$ `list(num)`
$nempty$	$[\![nempty]\!]=$`list(num)`
$(\mathbf{if}\ c\ t\ e)$	$[\![(\mathbf{if}\ c\ t\ e)]\!]=[\![t]\!]$ $[\![(\mathbf{if}\ c\ t\ e)]\!]=[\![e]\!]$ $[\![c]\!]=$`boolean`
$(\mathbf{lambda}\ (x)\ b)$	$[\![(\mathbf{lambda}\ (x)\ b)]\!]= [\![x]\!]\rightarrow[\![b]\!]$
$(f\ a)$	$[\![f]\!] = [\![a]\!]\rightarrow[\![(f\ a)]\!]$

Inferencia de tipos:

- Función factorial

Inferencia de tipos

```
(define fact
   (lambda (n)
      (cond [(zero? n)  1]
               [true   (*  n  (fact  (sub1 n)))])))
```

¿De qué elementos queremos inferir su tipo?

Inferencia de tipos: ¿cuántas sub-expresiones contiene?

```
(define fact
  (lambda (n)
    (cond [(zero? n)   1 ]
          [true   (*  n  (fact  (sub1 n)))]))))
```

Inferencia de tipos: ¿cuántas sub-expresiones contiene?

Nombramos a las sub-expresiones (con cuadritos y adentro el número de sub-expresión):

(**define** *fact*
  $\boxed{1}$(**lambda** (*n*)
    $\boxed{2}$(**cond**
      [$\boxed{3}$(*zero? n*) $\boxed{4}$1]
      [$\boxed{5}$true $\boxed{6}$(* *n* $\boxed{7}$(*fact* $\boxed{8}$(*sub1 n*)))])))

Derivamos la 1er. sub-expresión

$$[\![(\textbf{lambda}\ (x)\ b)]\!] = [\![x]\!] \rightarrow [\![b]\!]$$

$[\![\ \boxed{1}\ ]\!] = [\![$(lambda (n)

(cond [(zero? n) 1]

[true (* n (fact (sub1 n)))])) $]\!]$

$= [\![n]\!] \longrightarrow [\![$ (cond [(zero? n) 1]

[true (* n (fact (sub1 n)))]) $]\!]$

Usando los nombres de las sub-expresiones

$$= \quad [\![\texttt{n}]\!] \longrightarrow [\![\; \texttt{(cond [(zero? n) 1]} \quad \texttt{[true (* n (fact (sub1 n)))])} \;]\!]$$

$$= \quad [\![\boxed{1}]\!] = [\![n]\!] \rightarrow [\![\boxed{2}]\!]$$

Derivamos la 2da. sub-expresión

$[\![\boxed{2}]\!]$ = [[(cond [(zero? n) 1]

[true (* n (fact (sub1 n)))])]]

= [[(zero? n)]] regresa [[1]]

OR [[true]] regresa [[(* n (fact (sub1 n)))])]]

i.e. Recibe

$$[\![\boxed{3}]\!] = \mathit{boolean}$$
$$[\![\boxed{5}]\!] = \mathit{boolean}$$

y regresa

$$[\![\boxed{2}]\!] = [\![\boxed{4}]\!] = [\![\boxed{6}]\!]$$

Derivamos la 3era. sub-expresión $[\![(\mathit{zero?}\ e)]\!] = \mathtt{boolean} \quad [\![e]\!] = \mathtt{number}$

$[\![\,\boxed{3}\,]\!] = [\![\,(\text{zero?}\ \ n)\,]\!] = \text{boolean}$ y $[\![\,n\,]\!] = \text{number}$

Así que podemos decir que la primitiva zero? que es una función:

$$[\![n]\!] \rightarrow [\![\,\boxed{3}\,]\!] = \mathit{number} \rightarrow \mathit{boolean}$$

Derivamos la 4ta. sub-expresión

$[\![\boxed{4}]\!] = [\![1]\!] = \text{number}$

es decir, $[\![\boxed{4}]\!] = \mathit{number}$

IMPORTANTE: no es la expresión “cuadrito 1” es el número 1.

Derivamos la 5ta. sub-expresión

$[\![\boxed{5}]\!] = [\![\text{true}]\!] = \text{boolean}$

IMPORTANTE: es la constante true y en nuestras restricciones ya tiene asignado un tipo

Derivamos la 6ta. sub-expresión

$$[\![\boxed{6}]\!] = [\![\ (*\ \ n\ \ (\text{fact}\ \ (\text{sub1}\ \ n)))\ ]\!] = \text{number}$$

Es una multiplicación entonces:

$$[\![n]\!] \times [\![\boxed{7}]\!] \rightarrow [\![\boxed{6}]\!] = \mathit{number} \times \mathit{number} \rightarrow \mathit{number}$$

$[\![\ (*\ \ n\ \ (\text{fact}\ \ (\text{sub1}\ \ n)))\ ]\!] = \text{number}$ y sus argumentos

$$[\![\ n\ ]\!] = [\![\ (\text{fact}\ \ (\text{sub1}\ \ n))\ ]\!] = \text{number}$$

Derivamos la 7ma. sub-expresión

$$\llbracket f \rrbracket = \llbracket a \rrbracket \rightarrow \llbracket (f\ a) \rrbracket$$

$$\llbracket \boxed{7} \rrbracket = \llbracket \ (\texttt{fact}\ \ (\texttt{sub1}\ \ \texttt{n}))) \ \rrbracket = \llbracket \ (\texttt{fact}\ \boxed{8}\ \ ) \ \rrbracket$$

$$= \ \llbracket (\texttt{sub1}\ \ \texttt{n}) \rrbracket \longrightarrow \llbracket \ (\texttt{fact}\ \ (\texttt{sub1}\ \ \texttt{n}))) \ \rrbracket$$

$$\llbracket \boxed{1} \rrbracket = \llbracket \boxed{8} \rrbracket \rightarrow \llbracket \boxed{7} \rrbracket$$

Derivamos la 8va. sub-expresión

[[$\boxed{8}$]] = [[(sub1 n)]] = number y [[n]] = number

∴ el tipo de la función fact es:

number → number

donde [[n]] = number

Inferencia de tipos

- **Función nlength**

Función nlength

```
(define  nlength
     (lambda  ( l )
          (cond [ (nempty?   l )   0]
                    [ (ncons?   l )   (add1  (nlength  (nrest  l  )))])))
```

¿De qué tipo es la función?, ¿de qué tipo es su argumento (“ *l* ”) ?

Nombramos a las sub-expresiones:

(**define** *nlength*
  $\boxed{1}$(**lambda** (*l*)
    $\boxed{2}$(**cond**
      [$\boxed{3}$(*nempty? l*) $\boxed{4}$0]
      [$\boxed{5}$(*ncons? l*) $\boxed{6}$(*add1* $\boxed{7}$(*nlength* $\boxed{8}$(*nrest l*)))])))

Derivamos la 1er. sub-expresión

$[\![(\textbf{lambda}\ (x)\ b)]\!] = [\![x]\!] \rightarrow [\![b]\!]$

$[\![\boxed{1}]\!] = [\![$ (lambda (ι)

(cond [(nempty? ι) 0]

[(ncons? ι) (add1 (nlength (nrest ι)))])) $]\!]$

$= [\![\iota]\!] \longrightarrow [\![$ (cond [(nempty? ι) 0]

[(ncons? ι) (add1 (nlength (nrest ι)))]) $]\!]$

$= [\![\iota]\!] \longrightarrow [\![\boxed{2}]\!]$

Ahora la 2da. sub-expresión:

$[\![\boxed{2}]\!] = [\![$ (cond [(nempty? l) 0]

[(ncons? l) (add1 (nlength (nrest l)))]) $]\!]$

$= [\![$ (cond $[\![\boxed{3}]\!]$ REGRESA $[\![\boxed{4}]\!]$

OR $[\![\boxed{5}]\!]$ REGRESA $[\![\boxed{6}]\!]$) $]\!]$

Ahora la 3era. sub-expresión:

$$[\![\boxed{3}]\!] = [\![(\text{nemtpy?}\ \iota)]\!] = \text{boolean}$$

$$= [\![\iota]\!] \to \text{boolean} \quad \text{y} \quad [\![\iota]\!] = \text{nlist}$$

Ahora la 4ta. y 5ta. sub-expresiones:

$[\![\boxed{4}]\!] = [\![0]\!] = \text{number}$

$[\![\boxed{5}]\!] = [\![\text{(ncons?} \ \iota \text{)}]\!] = \text{boolean}$

$= [\![\iota]\!] \longrightarrow \text{boolean}$

$\text{donde } [\![\iota]\!] = \text{nlist}$

Ahora la 6ta. sub-expresión:

$[\![\ \boxed{6}\ ]\!] = [\![$ (add1 (nlength (nrest l))) $]\!] =$ number

y $[\![\ \boxed{7}\ ]\!] =$ number

Ahora derivamos la expresión 7

Ahora la 7ma. y 8va. sub-expresiones:

$$[\![f]\!] = [\![a]\!] \rightarrow [\![(f\ a)]\!]$$

$[\![\boxed{7}]\!] = [\![(\text{nlength}\ (\text{nrest}\ \ \iota))]\!]$ usando la restricción de tipo

$$[\![\text{nlength}]\!] = [\![\boxed{8}]\!] \rightarrow [\![(\text{nlength}\ \boxed{8})]\!]$$

$[\![\boxed{8}]\!] = [\![(\text{nrest}\ \ \iota)]\!] = \text{nlist}$

donde $[\![\iota]\!] = \text{nlist}$

Por lo tanto los tipos de la función son:

$\therefore$ **el tipo de la función nlength es:**

$$\textbf{nlist} \rightarrow \textbf{number}$$

donde $[\![\, \iota \,]\!] = \text{nlist}$

Inferencia de tipos:

- Función nlsum

Función: nlsum

```
(define nlsum
        (lambda ( lst )
                (cond  [ (nempty? lst)  0 ]
                        [(ncons?   lst)   (+  (nrest  lst) (nlsum (nrest  lst)))])))
```

¿De qué tipo es la función?, ¿los argumentos que recibe y lo que regresa?

1. Nombramos a las sub-expresiones

(**define** *nlsum*

$\boxed{1}$(**lambda** (*l*)

$\boxed{2}$(**cond**

[$\boxed{3}$(*nempty? l*) $\boxed{4}$0]

[$\boxed{5}$(*ncons? l*) $\boxed{6}$(+ $\boxed{7}$(*nrest l*)

$\boxed{8}$(*nlsum* $\boxed{9}$(*nrest l*)))])))

Derivamos la 1er. sub-expresión

$[\![(\textbf{lambda}\ (x)\ b)]\!] = [\![x]\!] \rightarrow [\![b]\!]$

$[\![\ \boxed{1}\ ]\!] = [\![$ (lambda (lst)

(cond [(nempty? lst) 0]

[(ncons? lst) (+ (nrest lst) (nlsum (nrest lst)))])) $]\!]$

$=\ [\![$ lst $]\!] \longrightarrow [\![$ (cond [(nemtpy? lst) 0]

[(ncons? lst) (+ (nrest lst) (nlsum (nrest lst)))])) $]\!]$

$= [\![$ lst $]\!] \longrightarrow [\![\ \boxed{2}\ ]\!]$

Ahora la 2da. sub-expresión:

⟦ $\boxed{2}$ ⟧ = ⟦ (cond [(nemtpy?   lst) 0]

[(ncons?   lst) (+   (nrest lst) (nlsum (nrest lst)))]))   ⟧

= ⟦ (cond   ⟦ $\boxed{3}$ ⟧   REGRESA ⟦ $\boxed{4}$ ⟧

OR   ⟦ $\boxed{5}$ ⟧   REGRESA ⟦ $\boxed{6}$ ⟧) ⟧

Ahora la 3era. sub-expresión:

$$[\![\boxed{3}]\!] = [\![\text{(nemtpy? lst)}]\!] = \text{boolean}$$

$$= [\![\text{lst}]\!] \rightarrow \text{boolean} \quad \text{y} \quad [\![\text{lst}]\!] = \text{nlist}$$

Ahora la 4ta. y 5ta. sub-expresiones:

$[\![\boxed{4}]\!] = [\![0]\!] = \text{number}$

$[\![\boxed{5}]\!] = [\![\text{(ncons? lst)}]\!] = \text{boolean}$

$= [\![\text{lst}]\!] \rightarrow \text{boolean}$

donde $[\![\text{lst}]\!] = \text{nlist}$

Ahora la 6ta. sub-expresión:

$[\![\ \boxed{6}\ ]\!] = [\![$ (+ (nrest lst) (nlsum (nrest lst))) $]\!] =$ number

$= [\![\ \boxed{7}\ ]\!] \times [\![\ \boxed{8}\ ]\!] =$ number x number

Y al derivar tanto la expresión 7 como 8 se debe de mantener que sean de tipo number, pues es una restricción sobre la operación suma.

Ahora derivemos 7 y después 8

Ahora la 7ma. y 8va. sub-expresiones:

$[\![\boxed{7}]\!] = [\![\text{(nrest lst)}]\!] = \text{nlist}$

y $[\![\text{lst}]\!] = \text{nlist}$

Y teníamos $[\![\boxed{7}]\!] = \text{number}$

(de la sub-expresión 6) !

¿Dudas?

Gracias