Fundamentos de lenguajes de programación

Semántica de los Conceptos Fundamentales de Lenguajes de Programación: Asignación

Facultad de Ingeniería. Universidad del Valle

Enero de 2021

Universidad del Valle

Contenido

Universidad del Valle

Contenido

Universidad del Valle

Asignación de variables

- Hasta el momento, solo hemos considerado el valor producido por una computación.
- No obstante, una computación puede también tener efectos: en ella se puede leer, imprimir o alterar el estado de la memoria o de un archivo del sistema, etc.
- La diferencia entre producir un valor y producir un efecto es que un efecto es global, esto es, un efecto afecta toda la computación.
- Nosotros nos concentraremos principalmente en un único efecto, la asignación de ubicaciones en memoria.

Asignación de variables

Diferencias entre ligadura y asignación:

- La ligadura de una variable es una acción local, mientras que la asignación de una variable es potencialmente global.
- Una ligadura crea una nueva asociación de un nombre con un valor, mientras que la asignación cambia el valor de una ligadura existente.
- La ligadura comprende la asociación de nombres con valores; asignación comprende el compartimiento de valores entre diferentes procedimientos.

Universidad del Valle

Asignación de variables

- Hasta el momento, las expresiones del lenguaje realizan alguna operación y retornan un valor expresado.
- No obstante, para permitir la asignación de variables es necesario permitir la ejecución secuencial de expresiones.

Asignación de variables

- Nuestro lenguaje será extendido para incorporar ejecución secuencial de expresiones y asignación de variables.
- El lenguaje consistirá de las expresiones especificadas anteriormente y de expresiones para ejecución secuencial `begin ...; ... end` y asignación de variables `set ... = ...`.
- Para este lenguaje se extiende el conjunto de valores expresados y denotados de la siguiente manera:

$$\begin{aligned} \text{Valor Expresado} &= \text{Número} + \text{Booleano} + \text{ProcVal} \\ \text{Valor Denotado} &= \text{Ref(Valor Expresado)} \end{aligned}$$

Universidad del Valle

Asignación de variables

Gramática

Se añaden las siguientes producciones a la gramática:

$$\langle \text{expresión} \rangle \quad ::= \quad \texttt{set}\ \langle \text{identificador} \rangle = \langle \text{expresión} \rangle$$

```
varassign-exp (ids rhs-exp)
```

$$::= \quad \texttt{begin}\ \langle \text{expresión} \rangle\ \{\texttt{;}\ \langle \text{expresión} \rangle\}^*\ \texttt{end}$$

```
begin-exp (exp exps)
```

Universidad del Valle

Asignación de variables

Se deben añadir las siguientes producciones a la especificación de la gramática:

```
    (expression ("begin" expression (arbno ";" expression) "
        end")
        begin-exp)
    (expression ("set" identifier "=" expression)
        set-exp)
```

Asignación de variables

De esta manera se puedan crear programas como:

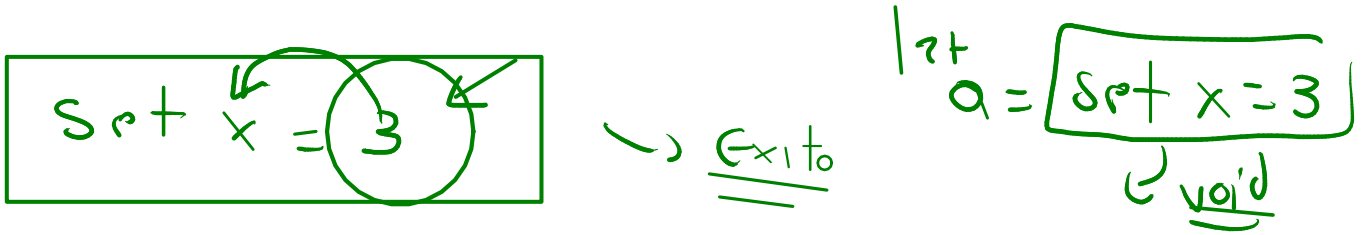
```
let m = 0
in
 begin
  set m = add1(m);
  set m = *(m, 2);
  m
 end

let x = 100
in
  let p = proc (x)
            begin
              set x = add1(x);
              x
            end
  in
    +((p x), (p x))
```

Semántica de la asignación de variables

- Para determinar el valor de una expresión `begin(exp exps)` se debe evaluar la expresión `exp` y cada una de las expresiones `exps`.
- Si `exps` es una lista vacía de expresiones se debe retornar el valor de la expresión `exp`. En caso contrario, se debe retornar el valor de la última expresión en `exps`.

Universidad del Valle

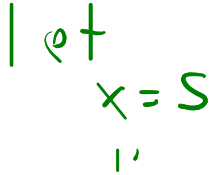
Semántica de la asignación de variables



- Para evaluar una expresión de asignación de variables (`set`) se debe evaluar la expresión de la parte derecha de la asignación.
- Luego, se debe modificar el contenido correspondiente a la variable con identificador igual a la parte izquierda de la asignación por este valor.
- El resultado de la expresión de asignación original es cualquier valor símbolico dado que esta expresión solo causa un efecto pero no produce un valor.

Semántica de la asignación de variables

- Un ambiente es un tipo de dato que asigna valores a variables.
- Cuando se incluyeron los procedimientos, se encontró el problema de que el estado del ambiente no era guardado en el momento de la aplicación de procedimientos y esto ocasionaba el retorno de un valor errado.



Semántica de la asignación de variables

- Con la inclusión de la asignación de variables surge otro problema. Cuando se crea un procedimiento se guarda el estado del ambiente, por esta razon al hacer un llamado al procedimiento, este se ejecuta sobre el ambiente que tiene almacenado y por ende no es sensible a los cambios en las variables del ambiente.

Semántica de la asignación de variables

```
let z = 0
    in
       let
        f = proc (x) z
           in
              begin
                 set z = 1;
                 (f 2)
              end
```

- La expresión anterior tiene como valor 0, en vez de 1.
- Esto se debe a que en el momento de la asignación de z a 1, el valor de dicha variable en el ambiente no cambió; lo que cambió fué el valor de la asociación creada en la ligadura local.
- Luego en el llamado al procedimiento, la evaluación se realiza en el ambiente almacenado (donde z tiene el valor de 0), por lo que se retorna 0.

Semántica de la asignación de variables

- Para evitar el problema anterior, cada identificador debe denotar la dirección de una ubicación en memoria (la memoria es también llamada *store*).
- Dicha dirección se denomina referencia y lo que hace la asignación es modificar su contenido.
- Las referencias o ubicaciones son tambien llamadas *L-valores* (valores que se asocian con variables que aparecen al lado izquierdo (*Left* en inglés) de la declaración de asignación).
- De forma análoga, los valores expresados (que aparecen en el lado derecho (*Right*) de la declaración de asignación) son llamados *R-valores*.

Semántica de la asignación de variables

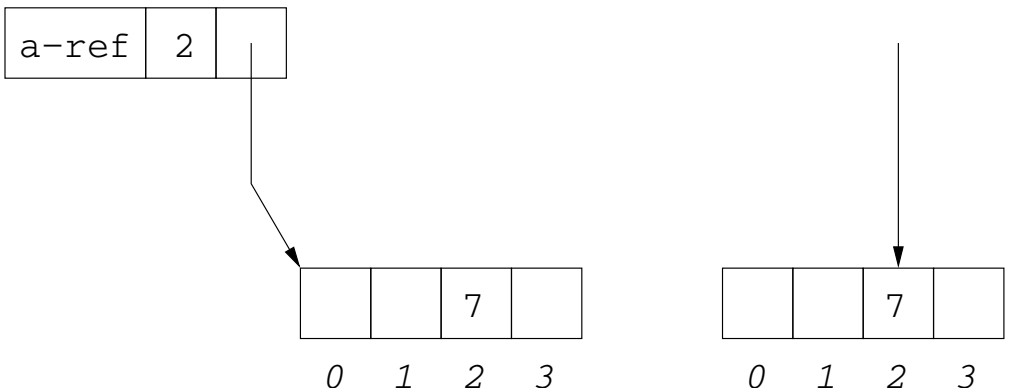
- Una referencia es un tipo de dato que contiene dos campos: un entero y un vector.
- El entero corresponde a la posición en el vector del valor asociado a la referencia.
- La interfaz del tipo de dato referencia consta de un procedimiento constructor y dos procedimientos observadores `deref` y `setref!`.

Semántica de la asignación de variables

- `a-ref (n v)`: crea una referencia.
- `deref (r)`: retorna el valor almacenado en la referencia.
- `setref!(r v)`: cambia el valor almacenado por la referencia.

Semántica de la asignación de variables

La siguiente figura muestra una referencia a una ubicación en un vector que contiene el valor 7.



Semántica de la asignación de variables

Definición de la interfaz del tipo de dato referencia:

```
(define-datatype reference reference?
  (a-ref (position integer?)
         (vec vector?)))
```

Semántica de la asignación de variables

```
(define deref
   (lambda (ref)
      (primitive-deref ref)))

(define primitive-deref
   (lambda (ref)
      (cases reference ref
         (a-ref (pos vec)
             (vector-ref vec pos)))))
```

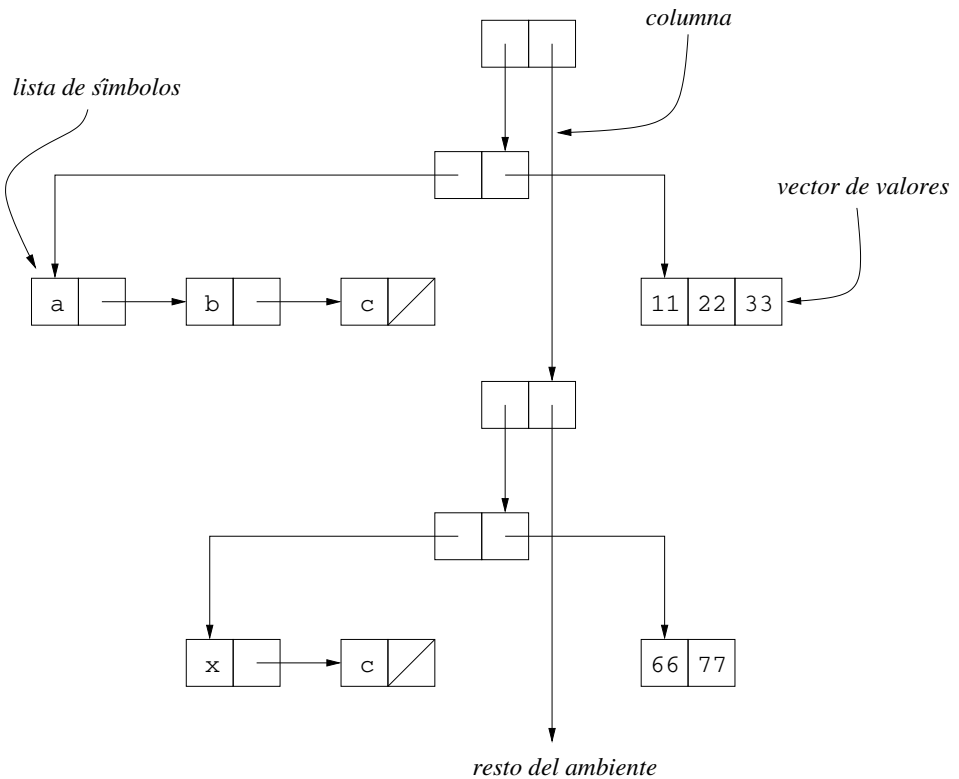
Universidad del Valle

Semántica de la asignación de variables

```
(define setref!
   (lambda (ref val)
      (primitive-setref! ref val)))

(define primitive-setref!
   (lambda (ref val)
      (cases reference ref
         (a-ref (pos vec)
             (vector-set! vec pos val)))))
```

Semántica de la asignación de variables

- Para implementar asignación, es necesario retomar el concepto de ambiente.
- Un ambiente puede verse como una estructura similar a la de la siguiente figura:



Universidad del Valle

Semántica de la asignación de variables

El tipo de dato ambiente está definido de la siguiente manera:

```
(define-datatype environment environment?
  (empty-env-record)
  (extended-env-record
    (syms (list-of symbol?))
    (vec vector?)
    (env environment?)))
```

Semántica de la asignación de variables

```
(define empty-env
  (lambda ()
    (empty-env-record)))

(define extend-env
  (lambda (syms vals env)
    (extended-env-record syms (list->vector vals) env)))
```

Semántica de la asignación de variables

```
(define extend-env-recursively
  (lambda (proc-names idss bodies old-env)
    (let ((len (length proc-names)))
      (let ((vec (make-vector len)))
        (let ((env (extended-env-record proc-names vec
                     old-env)))
          (for-each
            (lambda (pos ids body)
              (vector-set! vec pos (closure ids body env)))
            (iota len) idss bodies)
          env)))))
```

Semántica de la asignación de variables

```
(define iota
  (lambda (end)
    (let loop ((next 0))
      (if (>= next end) '()
        (cons next (loop (+ 1 next)))))))
```

Semántica de la asignación de variables

- Así mismo, se incluirá la operación `apply-env-ref` a la interfaz de ambiente para que cuando se encuentre un identificador, se retorne la referencia en vez de su valor.
- El procedimiento `apply-env` se reescribirá en términos de `apply-env-ref` y `deref`.

Semántica de la asignación de variables

```
(define apply-env
  (lambda (env sym)
    (deref (apply-env-ref env sym))))

(define apply-env-ref
  (lambda (env sym)
    (cases environment env
      (empty-env-record ()
        (eopl:error 'apply-env-ref "No binding for ~s" sym))
      (extended-env-record (syms vals env)
        (let ((pos (rib-find-position sym syms)))
          (if (number? pos)
            (a-ref pos vals)
            (apply-env-ref env sym)))))))
```

Semántica de la asignación de variables

El comportamiento de las expresiones de asignación de variables y de ejecución secuencial se obtiene agregando las siguientes clausulas en el procedimiento `eval-expression`:

```
(varassign-exp (id rhs-exp)
   (begin
     (setref!
       (apply-env-ref env id)
       (eval-expression rhs-exp env))
      1))
```

Universidad del Valle

Semántica de la asignación de variables

```
(begin-exp (exp exps)
   (let loop ((acc (eval-expression exp env))
              (exps exps))
        (if (null? exps)
            acc
            (loop (eval-expression (car exps) env)
                  (cdr exps))))))
```

Ejemplos

```
let
  x=2
in
  set x= 3
```

El valor de esta expresión es 1.

Ejemplos

```
let
  x=2
in
  begin
    set x= 3;
    x
  end
```

El valor de esta expresión es 3.

Ejemplos

```
let
  x = 100
in
  let
    p = proc (x)
          begin
            set x = add1(x);
            x
          end
  in
    +((p x), (p x))
```

El valor de esta expresión es 202.

Contenido

Universidad del Valle

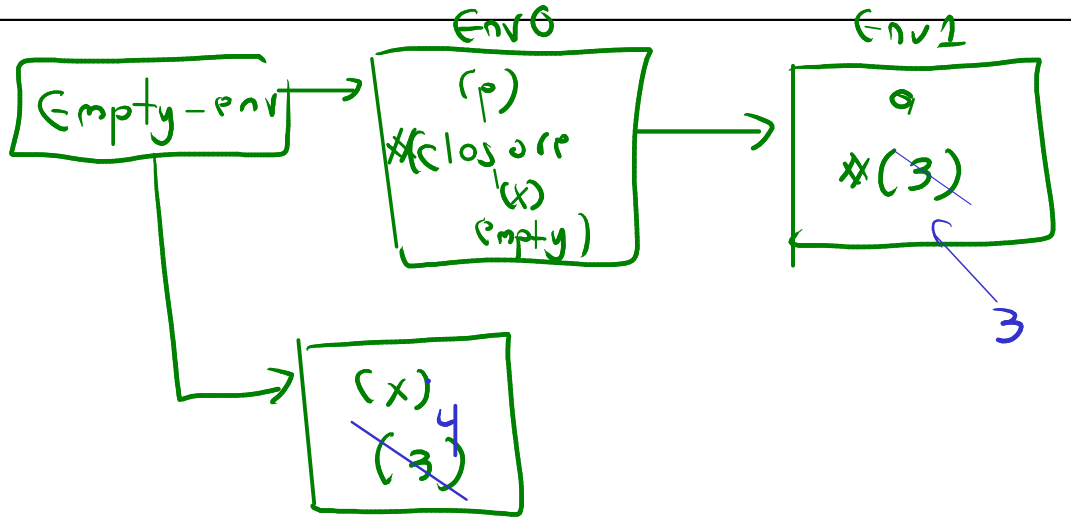
Paso de parámetros por valor

- En el lenguaje que se ha definido hasta el momento, en cada llamado a un procedimiento se crea una nueva referencia para cada parámetro formal.
- Esta política es conocida como *llamado o paso por valor*.
- Cuando se hace una asignación a un parámetro formal, la asignación es local al procedimiento.

Paso de parámetros por valor

Considere la expresión:

```
let
  p = proc(x)
        set x = 4
in
  let
    a = 3
  in
    begin
      (p a);
      a
    end
```



Paso de parámetros por valor

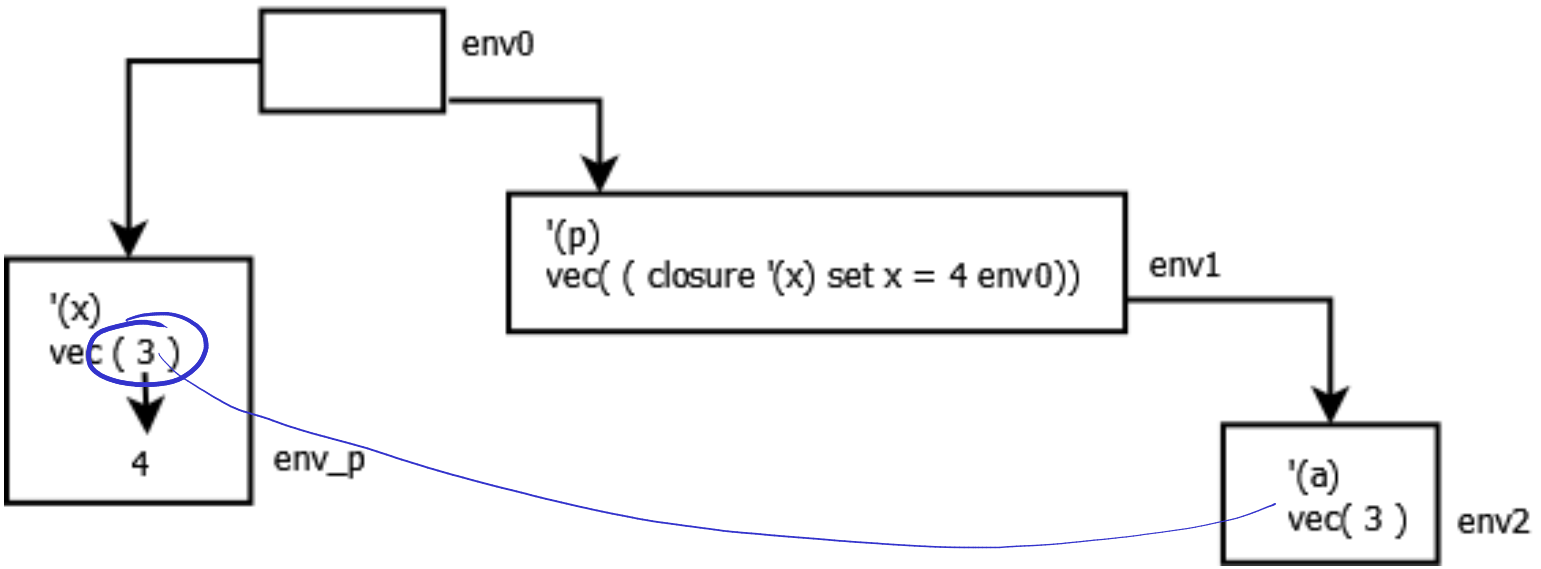
- Cuando se hace el llamado al procedimiento `p` (expresión `(p a)`) se evalua la expresión `a` y se crea una nueva referencia que contiene el valor de `a` (en un nuevo ambiente).
- La expresión de asignación en el cuerpo de `p` (la expresión `set x = 4`) se evalúa en el ambiente creado. Esta asignación afecta la nueva referencia que inicialmente almacena el valor 3.

Paso de parámetros por valor

- El nuevo contenido de la referencia interna en el procedimiento `p` es 4.
- No obstante, la última expresión de la expresión `begin` es la expresión `a`. Esta expresión se evalúa en el ambiente creado en el `let` que contiene la declaración de `a` y por ende su valor es 3.
- Finalmente, el valor de toda la expresión es 3.

Paso de parámetros por valor

Si se asume que la expresión anterior se evalúa en un ambiente env_0, la siguiente figura muestra los ambientes creados en la evaluación de esta expresión:



Universidad del Valle

Paso de parámetros por referencia

- Algunas veces es deseable que se permita pasar a un procedimiento variables con el objetivo de que éstas sean asignadas por dicho procedimiento.
- Eso significa que el valor de las variables cambia tanto en el interior del procedimiento como en el llamado al procedimiento.
- Lo anterior se realiza pasando al procedimiento una referencia a la ubicación de la variable y no su contenido.
- Este mecanismo es denominado *llamado o paso por referencia*.

Universidad del Valle

Paso de parámetros por referencia

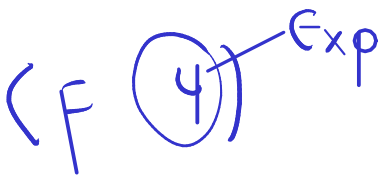
- Cuando se añade paso por referencia, los identificadores aún denotan referencias a valores expresados, luego los conjuntos no cambian:

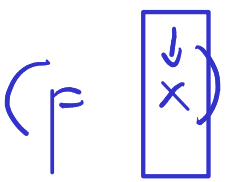
$$\begin{aligned} \text{Valor Expresado} &= \text{Número} + \text{Booleano} + \text{ProcVal} \\ \text{Valor Denotado} &= \text{Ref(Valor Expresado)} \end{aligned}$$

Paso de parámetros por referencia

Sin embargo, el cambio ocurre cuando se crean nuevas referencias.

- En los llamados por valor, una nueva referencia es creada para cada evaluacion de un operando.
- En los llamados por referencia, una nueva referencia es creada para cada evaluacion de un operando distinto a una variable.





Universidad del Valle

Paso de parámetros por referencia

- Para la implementación del llamado por referencia, una referencia será (como en llamado por valor) una pareja de una ubicación y un vector.
- La diferencia está en el contenido del vector, este puede ser:
  - Valores expresados (*blancos directos*)
  - Referencias a valores expresados (*blancos indirectos*)

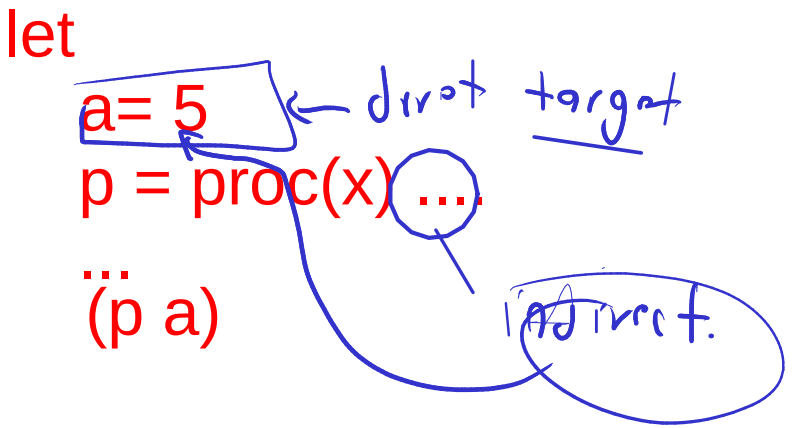
Paso de parámetros por referencia

- Blanco directo: El comportamiento del programa es igual al de paso por valor.
- Blanco indirecto: Corresponde al nuevo comportamiento del llamado por referencia, en el cual no son creadas nuevas ubicaciones.

Paso de parámetros por referencia

Un blanco (*target*) es un tipo de dato definido de la siguiente manera:

```
(define-datatype target target?
   (direct-target (expval expval?))
   (indirect-target (ref ref-to-direct-target?)))
```



Paso de parámetros por referencia

Los procedimientos correspondientes al tipo de dato `target` son:

- `expval?`: Retorna true si la entrada es un valor expresado, esto es, un número o un procedimiento.
- `ref-to-direct-target?`: Retorna true si la entrada es una referencia a un valor.

Paso de parámetros por referencia

```
(define expval?
   (lambda (x)
      (or (number? x) (procval? x))))

(define ref-to-direct-target?
   (lambda (x)
      (and (reference? x)
           (cases reference x
              (a-ref (pos vec)
                 (cases target (vector-ref vec pos)
                    (direct-target (v) #t)
                    (indirect-target (v) #f)))))))
```

Paso de parámetros por referencia

- La implementación (procedimientos `deref` y `setref!`) de la interfaz para el tipo de dato referencia cambian.
- Las nuevas definiciones observan el tipo de blanco almacenado en la referencia para determinar el valor expresado a retornar o la ubicación a cambiar.

Paso de parámetros por referencia

- La implementación del procedimiento `deref` tiene en cuenta que una referencia puede tener distintos blancos.
- Si el blanco es directo se retorna su valor.
- Si el blanco es indirecto (corresponde a una referencia a otra referencia) se busca el valor de la referencia interna.

Paso de parámetros por referencia

```
(define deref
   (lambda (ref)
      (cases target (primitive-deref ref)
         (direct-target (expval) expval)
         (indirect-target (ref1)
            (cases target (primitive-deref ref1)
               (direct-target (expval) expval)
               (indirect-target (p)
                  (eopl:error 'deref
                              "Illegal reference: ~s" ref1))
                              ))))))
```

Paso de parámetros por referencia

- La implementación del procedimiento `setref!` también tiene en cuenta que una referencia puede tener distintos blancos.
- Si el blanco es indirecto (corresponde a una referencia a otra referencia) se modifica el valor de la referencia interna.
- Si el blanco es directo se modifica la referencia de la entrada.
- En ambos casos el nuevo valor se almacena como un blanco directo.

Universidad del Valle

Paso de parámetros por referencia

```
(define setref!
   (lambda (ref expval)
      (let
         ((ref (cases target (primitive-deref ref)
                  (direct-target (expval1) ref)
                  (indirect-target (ref1) ref1))))
               (primitive-setref! ref (direct-target expval)
                  ))))
```

Paso de parámetros por referencia

- Para aplicaciones de primitivas, solo se necesita evaluar las subexpresiones y pasar los valores al procedimiento `apply-primitive`.
- Se cambia la clausula correspondiente en el procedimiento `eval-expression` por:

```
(primapp-exp (prim rands)
   (let ((args (eval-primapp-exp-rands rands env)))
      (apply-primitive prim args)))
```

Universidad del Valle

Paso de parámetros por referencia

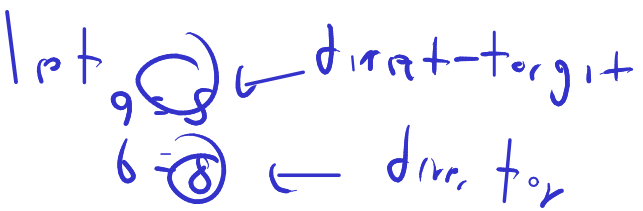
El procedimiento `eval-primapp-exp-rands` está definido de la siguiente manera

```
(define eval-primapp-exp-rands
  (lambda (rands env)
    (map (lambda (x) (eval-expression x env)) rands)))
```

Paso de parámetros por referencia

- Para ligadura local (expresiones `let`), se mantiene el comportamiento de los llamados por valor pero se hacen las modificaciones correspondientes para que los valores almacenados correspondan a blancos directos.
- Se cambia la clausula correspondiente en el procedimiento `eval-expression` por:

```
(let-exp (ids rands body)
    (let ((args (eval-let-exp-rands rands env)))
         (eval-expression body (extend-env ids args env
             ))))
```



Paso de parámetros por referencia

Los procedimientos `eval-let-exp-rands` y `eval-let-exp-rand` están definidos de la siguiente manera

```
(define eval-let-exp-rands
   (lambda (rands env)
        (map (lambda (x) (eval-let-exp-rand x env))
             rands)))

(define eval-let-exp-rand
   (lambda (rand env)
      (direct-target (eval-expression rand env))))
```

Paso de parámetros por referencia

- Para la aplicación de procedimientos, se evalúa cada operando usando el procedimiento `eval-rand`.
- Si el operando no es una variable, entonces se crea una nueva ubicación retornando el blanco directo.
- Si el operando es una variable, ésta denota una ubicación que contiene un valor expresado, luego se retorna un blanco indirecto que apunta a dicha ubicación.
- Si la variable está ligada a una ubicación que contiene un blanco directo, entonces una referencia a la ubicación es retornada. Pero si la variable está ligada a otra referencia, entonces dicha referencia es retornada.

Universidad del Valle

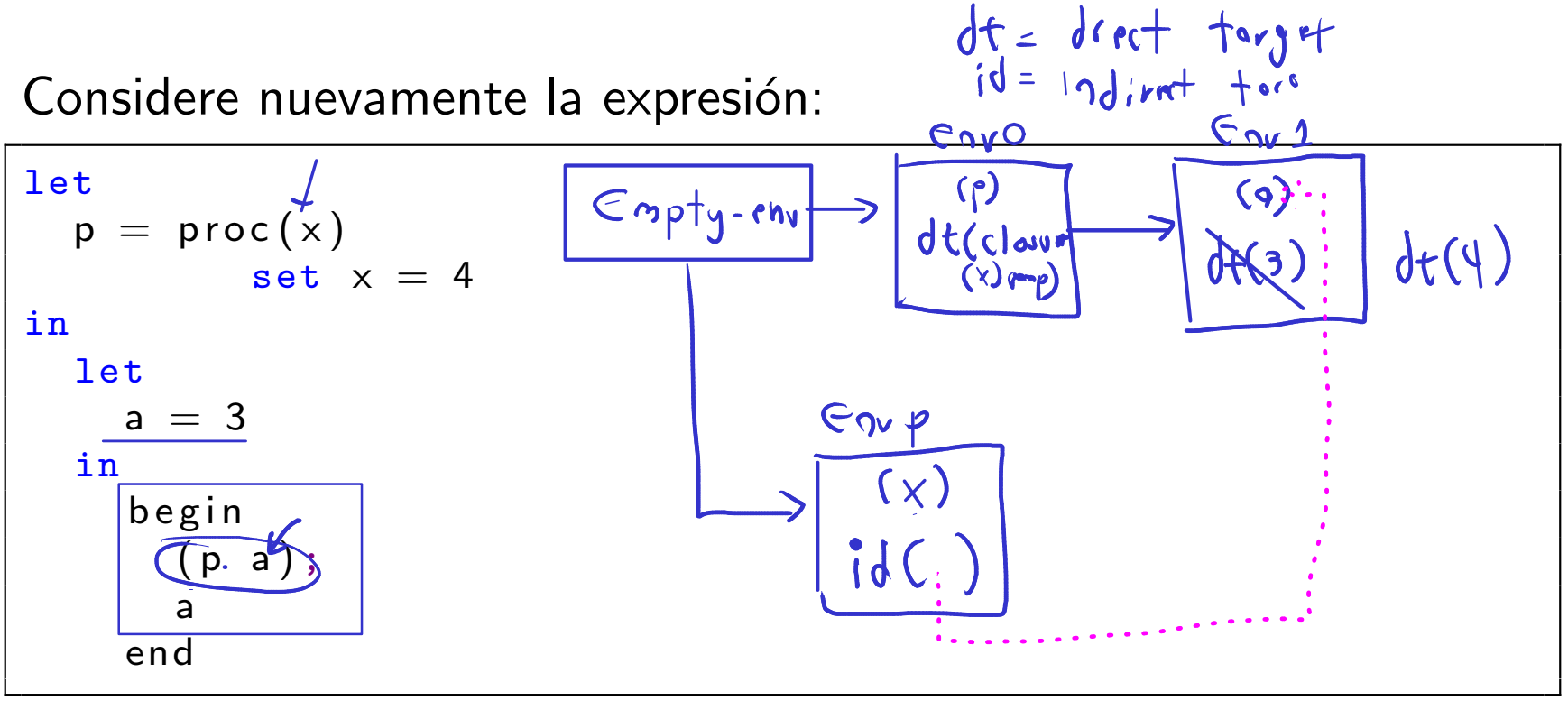
Paso de parámetros por referencia

Los procedimientos `eval-rand` está definido de la siguiente manera

```
(define eval-rand
   (lambda (rand env)
      (cases expression rand
         (var-exp (id)
            (indirect-target
               (let ((ref (apply-env-ref env id)))
                  (cases target (primitive-deref ref)
                     (direct-target (expval) ref)
                     (indirect-target (ref1) ref1)))))
         (else
            (direct-target (eval-expression rand env))))))
```

Paso de parámetros por referencia

Considere nuevamente la expresión:

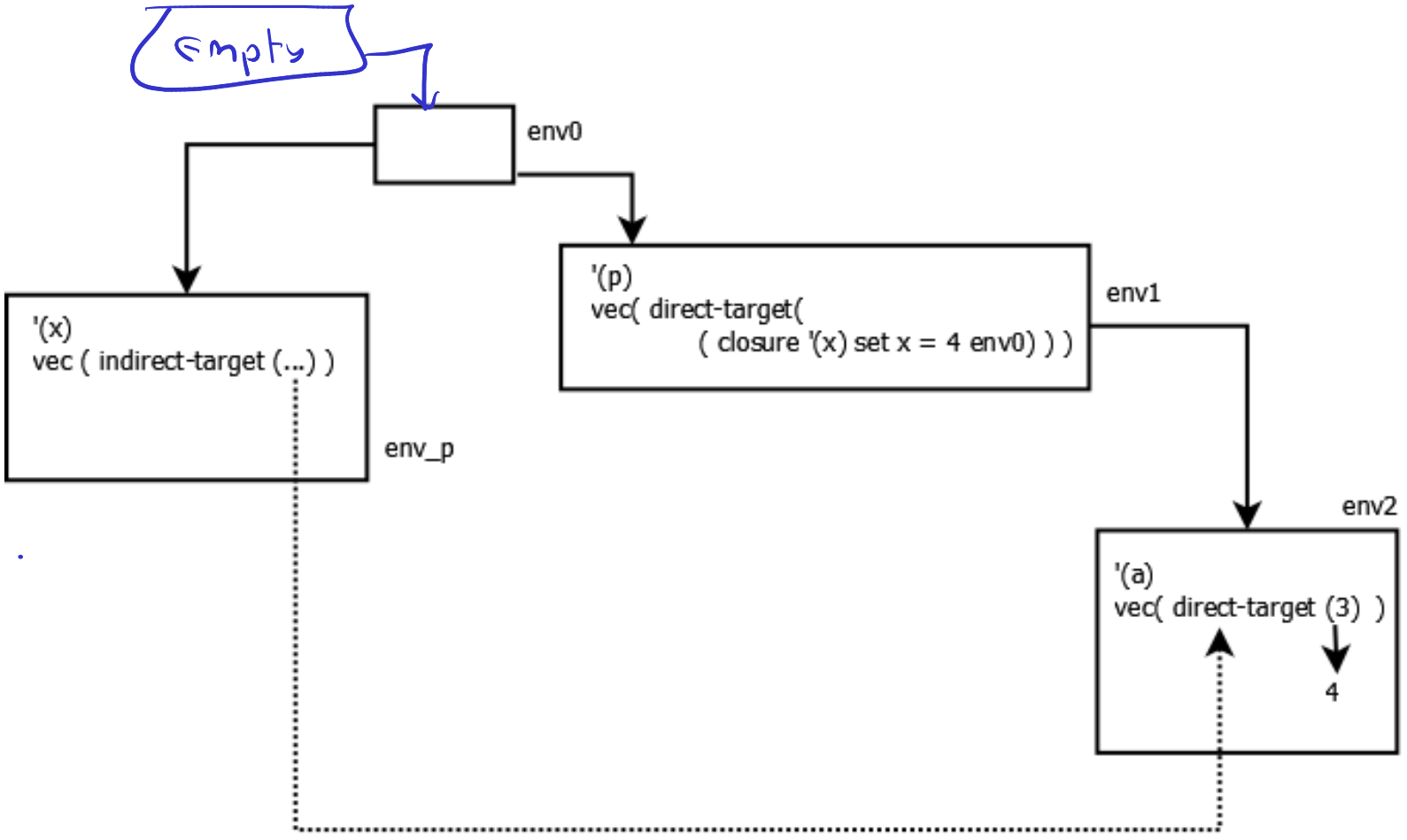


Con paso por referencia, el valor de esta expresión es 4.

Universidad del Valle

Paso de parámetros por valor

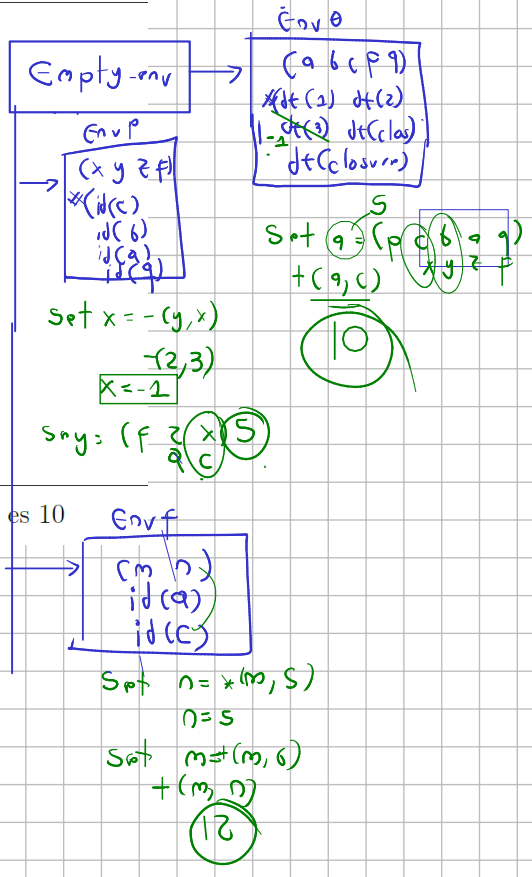
Luego, los ambientes creados en la evaluación de esta expresión se pueden visualizar así:



Universidad del Valle

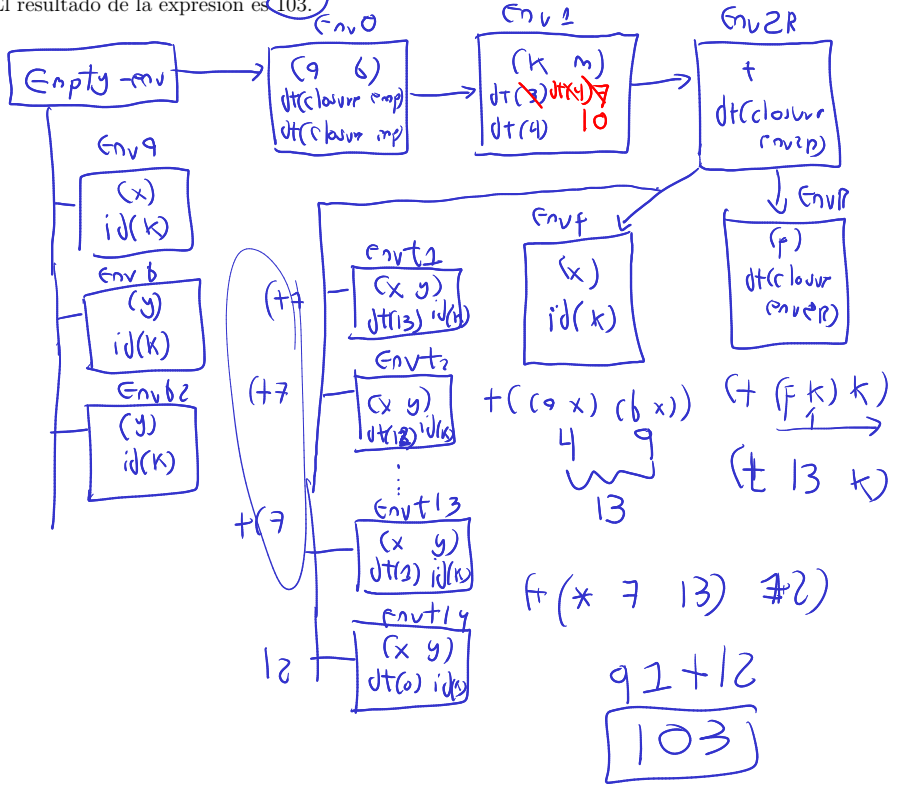
```
let
a = 1
b = 2
c = 3
p = proc(x, y, z, f)
      begin
          set x = -(y, x);
          set y = (f z x);
          x
      end

q = proc(m, n)
      begin
          set n = *(m, 5);
          set m = +(m, 6);
          +(m, n)
      end
in
      begin
          set a = (p c b a q);
          +(a, c)
      end
```

El resultado de la expresión es 10



```
let
  a = proc(x) begin set x = +(x,1); x end
  b = proc(y) begin set y = +(y,3); +(y,2) end
  in
     let
        k = 3
        m = 4
        in
        letrec
            t(x,y) = if >(x,0) then +(y, (t -(x,1) y)) else (b y)
            in
               let
                  f = proc(x) +( (a x), (b x))
                  in
                      (t (f k) k)
```

El resultado de la expresión es 103.



Preguntas

?

Universidad del Valle

Próxima sesión

- Tipos.