Fusión en presencia de acumuladores

Mónica Martínez

Facultad de Ingeniería
Universidad de la República
Pedeciba Informática

Maestria en Informática
Orientador: Alberto Pardo

Agenda

PROGRAMACIÓN POR COMPOSICIÓN DE FUNCIONES

$$
\begin{aligned}
&\mathit{factorial} && :: && \mathit{Num}\ a \Rightarrow a \rightarrow a \\
&\mathit{factorial} && = && \mathit{product} \circ \mathit{down}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{product} && :: && \mathit{Num}\ a \Rightarrow [a] \rightarrow a \\
&\mathit{product}\ [\,] && = && 1 \\
&\mathit{product}\ (a : as) && = && a * \mathit{product}\ as
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{down} && :: && \mathit{Num}\ a \Rightarrow a \rightarrow [a] \\
&\mathit{down}\ 0 && = && [\,] \\
&\mathit{down}\ n && = && n : \mathit{down}\ (n-1)
\end{aligned}
$$

Programación por composición de funciones

Ventajas vs. Desventajas

Ventajas

- Claridad
- Uso de bibliotecas
- Fácil mantenimiento

Desventajas

- Pérdida de eficiencia
  - Uso de estructuras intermedias – comunicación

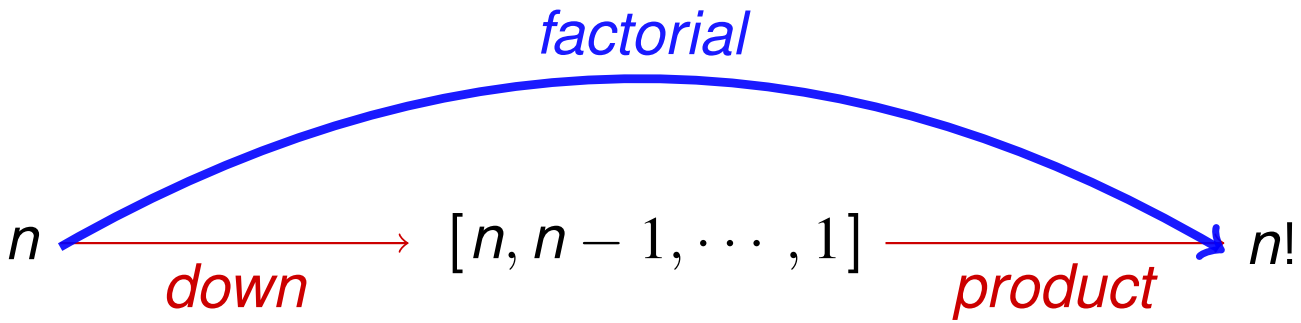
Alternativas

- Fusión

FUSIÓN DE FUNCIONES

$$n \xrightarrow[down]{} [n, n-1, \cdots, 1] \xrightarrow[product]{} n!$$

Fusión de funciones



- No se genera la estructura intermedia
- Menos uso del garbage collector
- Menos gerenciamiento de la memoria

PEDECIBA

Fusión de funciones

Definición:

$$\begin{aligned} \mathit{factorial}\ 0 &= 1 \\ \mathit{factorial}\ n &= n * \mathit{factorial}\ (n-1) \end{aligned}$$

Sin estructura intermedia

ACUMULACIONES

- Funciones recursivas que utilizan un parámetro adicional para el cálculo del resultado.
- Su uso es motivado por razones de eficiencia.
- Técnica de acumulaciones - transformación de definiciones recursivas en acumulaciones.

ACUMULACIONES

$$
\begin{aligned}
&\mathit{reverse} && :: && [a] \to [a] \\
&\mathit{reverse}\ [\,] && = && [\,] \\
&\mathit{reverse}\ (a : as) && = && \mathit{reverse}\ as +\!\!+ [a]
\end{aligned}
$$

ACUMULACIONES

$$
\begin{aligned}
&\mathit{reverse} && :: && [a] \to [a] \\
&\mathit{reverse}\ [\,] && = && [\,] \\
&\mathit{reverse}\ (a : as) && = && \mathit{reverse}\ as +\!\!+ [a]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{reverse}\ xs && = && \mathit{areverse}\ (xs, [\,]) \\
&\mathit{areverse} && :: && ([a], [a]) \to [a] \\
&\mathit{areverse}\ ([\,], xs) && = && xs \\
&\mathit{areverse}\ (a : as, xs) && = && \mathit{areverse}\ (as, a : xs)
\end{aligned}
$$

ACUMULACIONES
FUSIÓN

- Muchas técnicas clásicas presentan dificultades al considerar acumulaciones.
- No alcanzan los acumuladores en el caso de acumulaciones como productoras.

Fusión con Acumulaciones

Dada una lista de dígitos $d_n \cdots d_1 d_0$ convertirla en un número.
Ejemplo: $[5, 3, 7] \rightarrow 537$

$$
\begin{aligned}
number &:: [Int] \rightarrow Int \\
number\ ds &= anumber\ (ds, [\,]) \\
anumber\ (ds, xs) &= horner\ (areverse\ (ds, xs))
\end{aligned}
$$

$$
\begin{aligned}
horner\ [\,] &= 0 \\
horner\ (d : ds) &= d + 10 * horner\ ds
\end{aligned}
$$

$$
\begin{aligned}
areverse\ ([\,], xs) &= xs \\
areverse\ (a : as, xs) &= areverse\ (as, a : xs)
\end{aligned}
$$

Fusión con Acumulaciones (II)

Caso $[\,]$

$$
\begin{aligned}
& \mathit{anumber}\ ([\,], \mathit{xs}) \\
= & \quad \{\ \text{Definición } \mathit{anumber}\ \} \\
& \mathit{horner}\ (\mathit{areverse}\ ([\,], \mathit{xs})) \\
= & \quad \{\ \text{Def. de } \mathit{areverse}\ \} \\
& \mathit{horner}\ \mathit{xs}
\end{aligned}
$$

Fusión con Acumulaciones (III)

Caso $(a : as)$

$$
\begin{aligned}
& anumber\ (d : ds, xs) \\
= & \quad \{\ \text{Def. de } anumber\ \} \\
& horner\ (areverse\ (d : ds, xs)) \\
= & \quad \{\ \text{Def. de } areverse\ \} \\
& horner\ (areverse\ (ds, d : xs)) \\
= & \quad \{\ \text{Definición de } anumber\ \} \\
& anumber\ (ds, d : xs)
\end{aligned}
$$

Fusión con Acumulaciones (IV)

En resumen:

$$\begin{aligned} anumber\ ([\,], xs) &= horner\ xs \\ anumber\ (d : ds, xs) &= anumber\ (ds, d : xs) \end{aligned}$$

Fusión con Acumulaciones (IV)

En resumen:

$$\begin{aligned} anumber\ ([\,], xs) &= horner\ xs \\ anumber\ (d : ds, xs) &= anumber\ (ds, d : xs) \end{aligned}$$

Fusión con Acumulaciones (IV)

En resumen:

$$anumber\ ([\,], xs) = horner\ xs$$
$$anumber\ (d : ds, xs) = anumber\ (ds, d : xs)$$

ACUMULACIONES
FUSIÓN DE PROGRAMAS

- Problema abierto
- Diferentes abordajes
- Abordaje considerado: Short-Cut Fusion
  - Enfocado en el proceso de producción de la estructura intermedia
  - Sencillo, automatizable

SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1)
\end{aligned}
$$

$$
\begin{aligned}
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as
\end{aligned}
$$

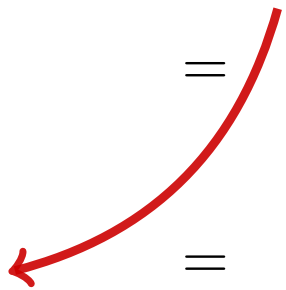
SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1)
\end{aligned}
$$

$$
\begin{aligned}
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as
\end{aligned}
$$

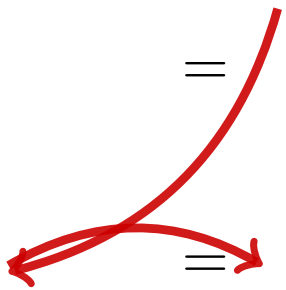
SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1) \\
\\
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as
\end{aligned}
$$



SHORT-CUT FUSION

$$
\begin{aligned}
&\mathit{down}\ 0 &&= [\,] \\
&\mathit{down}\ n &&= n : \mathit{down}\ (n-1) \\
&\\
&\mathit{product}\ [\,] &&= 1 \\
&\mathit{product}\ (a : as) &&= a * \mathit{product}\ as
\end{aligned}
$$



SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1) \\
\\
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as \\
\\
factorial\ 0 &= 1
\end{aligned}
$$

SHORT-CUT FUSION

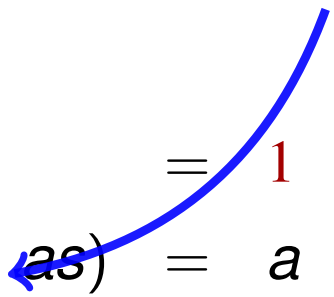
$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1)
\end{aligned}
$$

$$
\begin{aligned}
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as
\end{aligned}
$$

$$
factorial\ 0 = 1
$$

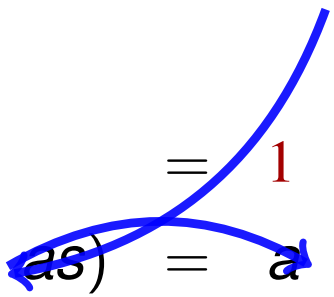
SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1) \\
\\
product\ [\,] &= 1 \\
product\ (a : as) &= a\ *\ product\ as \\
\\
factorial\ 0 &= 1
\end{aligned}
$$



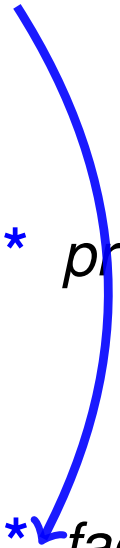
SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1) \\
\\
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as \\
\\
factorial\ 0 &= 1
\end{aligned}
$$



SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1) \\
\\
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as \\
\\
factorial\ 0 &= 1 \\
factorial\ n &= n * factorial\ (n-1)
\end{aligned}
$$



SHORT-CUT FUSION

$$
\begin{aligned}
down\ 0 &= [\,] \\
down\ n &= n : down\ (n-1)
\end{aligned}
$$

$$
\begin{aligned}
product\ [\,] &= 1 \\
product\ (a : as) &= a * product\ as
\end{aligned}
$$

$$
\begin{aligned}
factorial\ 0 &= 1 \\
factorial\ n &= n * factorial\ (n-1)
\end{aligned}
$$

SHORT-CUT FUSION

Consumidora - operador *fold*

$$
\begin{aligned}
&\mathit{fold} && :: && (b, a \to b \to b) \to [a] \to b \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ [\,] && = && \mathit{fn} \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ (a : \mathit{as}) && = && \mathit{fc}\ a\ (\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ \mathit{as})
\end{aligned}
$$

PEDECIBA

SHORT-CUT FUSION

Consumidora - operador *fold*

$$
\begin{aligned}
&\mathit{fold} && :: && (b, a \to b \to b) \to [a] \to b \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ [\,] && = && \mathit{fn} \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ (a : \mathit{as}) && = && \mathit{fc}\ a\ (\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ \mathit{as})
\end{aligned}
$$

$$
\mathit{fold}\ (z, (\oplus))\ [x_1, x_2, \cdots, x_n] = x_1 \oplus (x_2 \oplus (\cdots (x_n \oplus z) \cdots))
$$

SHORT-CUT FUSION

Consumidora - operador *fold*

$$
\begin{aligned}
&\mathit{fold} && :: && (b, a \to b \to b) \to [a] \to b \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ [\,] && = && \mathit{fn} \\
&\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ (a : \mathit{as}) && = && \mathit{fc}\ a\ (\mathit{fold}\ (\mathit{fn}, \mathit{fc})\ \mathit{as})
\end{aligned}
$$

$$
\mathit{fold}\ (z, (\oplus))\ [x_1, x_2, \cdots, x_n] = x_1 \oplus (x_2 \oplus (\cdots (x_n \oplus z) \cdots))
$$

$$
\begin{aligned}
\mathit{product}\ [x_1, x_2, \cdots, x_n] &= x_1 * (x_2 * (\cdots (x_n * 1) \cdots)) \\
&= \mathit{fold}\ (1, (*))
\end{aligned}
$$

SHORT-CUT FUSION

Productora - función *build*

$$build :: (\forall\, b\, .\, (b, a \to b \to b) \to c \to b) \to c \to [a]$$
$$build\ g = g\ ([\,], (:))$$

SHORT-CUT FUSION

Productora - función *build*

$$build :: (\forall\, b\, .\, (b, a \to b \to b) \to c \to b) \to c \to [a]$$
$$build\ g = g\ ([\,], (:))$$

$$down\ 0 = [\,]$$
$$down\ n = n : down\ (n-1)$$

SHORT-CUT FUSION

Productora - función *build*

$$build :: (\forall\, b\, .\, (b, a \to b \to b) \to c \to b) \to c \to [a]$$
$$build\ g = g\ ([\,], (:))$$

$$down = build\ gdown$$

where

$$gdown\ (fn, fc)\ 0 = fn$$
$$gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1))$$

Short-Cut Fusion

Regla Fold/Build

$$g :: \forall\ b\ .\ (b, a \to b \to b) \to c \to b$$

$$\Rightarrow$$

$$fold\ (fn, fc) \circ build\ g = g\ (fn, fc)$$

SHORT-CUT FUSION

$$
\begin{aligned}
product &= fold\ (1, (*)) \\
down &= build\quad gdown \\
&\textbf{where} \\
&\quad gdown\ (fn, fc)\ 0 = fn \\
&\quad gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1))
\end{aligned}
$$

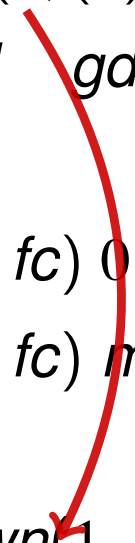
$$
product \circ down =
$$

SHORT-CUT FUSION

$$
\begin{aligned}
product &= fold\ (1, (*)) \\
down &= build\ \ gdown \\
&\textbf{where} \\
&\quad gdown\ (fn, fc)\ 0 = fn \\
&\quad gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1)) \\
\\
product \circ down &= gdown
\end{aligned}
$$

SHORT-CUT FUSION

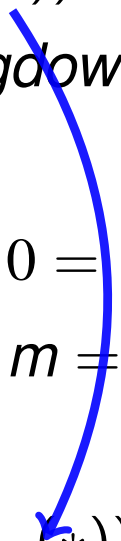
$$
\begin{aligned}
product &= fold\ (1, (*)) \\
down &= build\ \ gdown \\
&\textbf{where} \\
&\quad gdown\ (fn, fc)\ 0 = fn \\
&\quad gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1)) \\
\\
product \circ down &= gdown\ (1,
\end{aligned}
$$



PEDECIBA

SHORT-CUT FUSION

$$
\begin{aligned}
product &= fold\ (1, (*)) \\
down &= build\ \ gdown \\
&\textbf{where} \\
&\quad gdown\ (fn, fc)\ 0 = fn \\
&\quad gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1)) \\
\\
product \circ down &= gdown(1, (*))
\end{aligned}
$$



SHORT-CUT FUSION

$$
\begin{aligned}
\mathit{product} &= \mathit{fold}\ (1, (*)) \\
\mathit{down} &= \mathit{build}\quad \mathit{gdown} \\
&\textbf{where} \\
&\quad \mathit{gdown}\ (\mathit{fn}, \mathit{fc})\ 0 = \mathit{fn} \\
&\quad \mathit{gdown}\ (\mathit{fn}, \mathit{fc})\ m = \mathit{fc}\ m\ (\mathit{gdown}\ (\mathit{fn}, \mathit{fc})\ (m-1))
\end{aligned}
$$

$$
\mathit{product} \circ \mathit{down} = \mathit{gdown}(1, (*))
$$

$$
\begin{aligned}
\mathit{factorial}\quad 0 &= 1 \\
\mathit{factorial}\quad n &= n * \mathit{factorial}\ (n-1)
\end{aligned}
$$

ACUMULACIONES COMO CONSUMIDORAS

$$factorial\ m = aproduct\ (down\ m)\ 1$$

$$\begin{aligned} down &= build\ gdown \\ &\textbf{where} \\ &\quad gdown\ (fn, fc)\ 0 = fn \\ &\quad gdown\ (fn, fc)\ m = fc\ m\ (gdown\ (fn, fc)\ (m-1)) \end{aligned}$$

$$\begin{aligned} aproduct &:: Num\ a \Rightarrow [a] \rightarrow a \rightarrow a \\ aproduct\ [\,]\ n &= n \\ aproduct\ (a : as)\ n &= aproduct\ as\ (a * n) \end{aligned}$$

ACUMULACIONES COMO CONSUMIDORAS

$$
\begin{aligned}
aproduct\ [\,]\ n &= n \\
aproduct\ (a : as)\ n &= aproduct\ as\ (a * n)
\end{aligned}
$$

ACUMULACIONES COMO CONSUMIDORAS

$$
\begin{aligned}
aproduct\ [\,]\ n &= n \\
aproduct\ (a : as)\ n &= aproduct\ as\ (a * n)
\end{aligned}
$$

$$
\begin{aligned}
aproduct\ [\,] &= \lambda n \rightarrow n \\
aproduct\ (a : as) &= \lambda n \rightarrow aproduct\ as\ (a * n)
\end{aligned}
$$

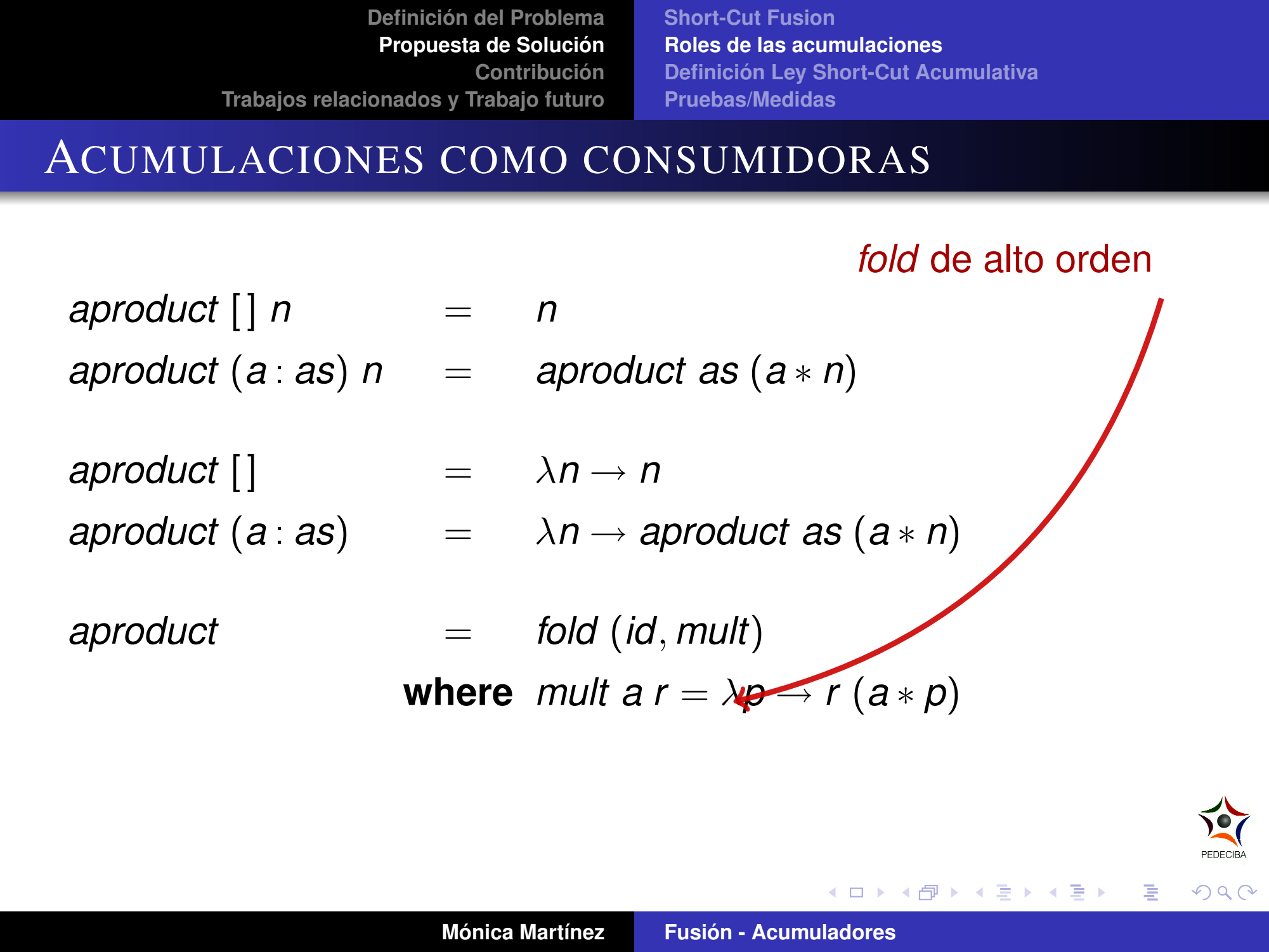
ACUMULACIONES COMO CONSUMIDORAS

$$
\begin{array}{lcl}
aproduct\ [\,]\ n & = & n \\
aproduct\ (a : as)\ n & = & aproduct\ as\ (a * n) \\
 & & \\
aproduct\ [\,] & = & \lambda n \rightarrow n \\
aproduct\ (a : as) & = & \lambda n \rightarrow aproduct\ as\ (a * n) \\
 & & \\
aproduct & = & fold\ (id, mult) \\
 & \textbf{where} & mult\ a\ r = \lambda p \rightarrow r\ (a * p)
\end{array}
$$

ACUMULACIONES COMO CONSUMIDORAS

fold de alto orden

$$
\begin{array}{lcl}
aproduct\ [\,]\ n & = & n \\
aproduct\ (a : as)\ n & = & aproduct\ as\ (a * n) \\
 & & \\
aproduct\ [\,] & = & \lambda n \rightarrow n \\
aproduct\ (a : as) & = & \lambda n \rightarrow aproduct\ as\ (a * n) \\
 & & \\
aproduct & = & fold\ (id, mult) \\
 & \textbf{where} & mult\ a\ r = \lambda p \rightarrow r\ (a * p)
\end{array}
$$



ACUMULACIONES COMO CONSUMIDORAS

fold de alto orden

$$
\begin{array}{lcl}
aproduct\ [\,]\ n & = & n \\
aproduct\ (a : as)\ n & = & aproduct\ as\ (a * n) \\
 & & \\
aproduct\ [\,] & = & \lambda n \rightarrow n \\
aproduct\ (a : as) & = & \lambda n \rightarrow aproduct\ as\ (a * n) \\
 & & \\
aproduct & = & fold\ (id, mult) \\
 & \textbf{where} & mult\ a\ r = \lambda p \rightarrow r\ (a * p)
\end{array}
$$

Short-Cut Fusion Standard

ACUMULACIONES COMO CONSUMIDORAS

$$
\begin{aligned}
\mathit{factorial}\ m &= \mathit{aproduct}\ (\mathit{down}\ m)\ 1 \\
&= \mathit{fold}\ (\mathit{id}, \mathit{mult})\ (\mathit{build}\ \mathit{gdown}\ m)\ 1
\end{aligned}
$$

ACUMULACIONES COMO CONSUMIDORAS

$$
\begin{aligned}
\mathit{factorial}\ m &= \mathit{aproduct}\ (\mathit{down}\ m)\ 1 \\
&= \mathit{fold}\ (\mathit{id}, \mathit{mult})\ (\mathit{build}\ \mathit{gdown}\ m)\ 1 \\
&= \mathit{gdown}\ (\mathit{id}, \mathit{mult})\ m\ 1
\end{aligned}
$$

Se define: $\mathit{afact} = \mathit{gdown}\ (\mathit{id}, \mathit{mult})$

$$
\begin{aligned}
\mathit{afact}\ \ 0 &= \mathit{id} \\
\mathit{afact}\ \ m &= \mathit{afact}\ (m-1) \circ (m*)
\end{aligned}
$$

Lista de llamadas pendientes

ACUMULACIONES COMO PRODUCTORAS

1. Acumulador como contexto
2. Acumulador como parte de la estructura
3. Mixto

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - CONTEXTO

$$pisum :: ([Integer], Integer) \rightarrow Integer$$

$$pisum = product \circ isum$$

isum - Devuelve la suma de los prefijos de una lista

$isum\ ([1,2,3],0) = [0,1,3,6]$

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - CONTEXTO

$$pisum :: ([Integer], Integer) \rightarrow Integer$$
$$pisum = product \circ isum$$

isum - Devuelve la suma de los prefijos de una lista
$isum\ ([1,2,3],0) = [0,1,3,6]$

$$\begin{aligned} isum &:: Num\ a \Rightarrow ([a], a) \rightarrow [a] \\ isum\ ([\,], s) &= [s] \\ isum\ (a : as, s) &= s : isum\ (as, a + s) \end{aligned}$$

$$product = fold\ (1, (*))$$

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - CONTEXTO

$$
\begin{aligned}
isum &= build\ gism \\
&\textbf{where} \quad gism\ (fn, fc)\ ([\,], s) = fc\ s\ fn \\
&\qquad\quad gism\ (fn, fc)\ (a : as, s) = fc\ s\ (gism\ (fn, fc)\ (as, a + s))
\end{aligned}
$$

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - CONTEXTO

$$
\begin{aligned}
isum &= build\ gism \\
&\textbf{where} \quad gism\ (fn, fc)\ ([\,], s) = fc\ s\ fn \\
&\qquad\quad gism\ (fn, fc)\ (a : as, s) = fc\ s\ (gism\ (fn, fc)\ (as, a + s))
\end{aligned}
$$

$$pisum = product \circ isum$$

$$pisum = fold\ (1, (*)) \circ build\ gism$$

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - CONTEXTO

$$
\begin{aligned}
isum &= build\ gism \\
&\textbf{where} \quad gism\ (fn, fc)\ ([\,], s) = fc\ s\ fn \\
&\qquad\quad gism\ (fn, fc)\ (a : as, s) = fc\ s\ (gism\ (fn, fc)\ (as, a + s))
\end{aligned}
$$

$$pisum = product \circ isum$$

$$pisum = fold\ (1, (*)) \circ build\ gism$$

Short-Cut Fusion Standard

Acumulaciones como productoras

Acumulador - contexto

$$
\begin{aligned}
isum &= build\ gism \\
&\textbf{where} \quad gism\ (fn, fc)\ ([\,], s) = fc\ s\ fn \\
&\qquad\quad gism\ (fn, fc)\ (a : as, s) = fc\ s\ (gism\ (fn, fc)\ (as, a + s))
\end{aligned}
$$

$$
\begin{aligned}
pisum &= product \circ isum \\
pisum &= fold\ (1, (*)) \circ build\ gism \\
&= gism\ (1, (*))
\end{aligned}
$$

Short-Cut Fusion Standard

$$
\begin{aligned}
pisum\ ([\,], s) &= s \\
pisum\ (a : as, s) &= s * pisum\ (as, a + s)
\end{aligned}
$$

Acumulaciones como productoras

Acumulador - Parte de la estructura

$$number\ ds = anumber\ (ds, [\,])$$

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$anumber\ ([\,], xs) = horner\ xs$$

$$anumber\ (d : ds, xs) = anumber\ (ds, d : xs)$$

Acumulaciones como productoras

Acumulador - Parte de la estructura

$$number\ ds = anumber\ (ds, [\,])$$

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$anumber\ ([\,], xs) = horner\ xs$$

$$anumber\ (d : ds, xs) = anumber\ (ds, d : xs)$$

Acumulaciones como productoras

Acumulador - Parte de la estructura

$$number\ ds = anumber\ (ds, [\,])$$

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$anumber\ ([\,], xs) = horner\ xs$$

$$anumber\ (d : ds, xs) = anumber\ (ds, d : xs)$$

Def. esperada

$$anumber\ (ds, xs) = anumber'\ (ds, horner\ xs)$$

$$anumber'\ ([\,], z) = z$$

$$anumber'\ (a : as, z) = anumber'\ (as, a + 10 * z)$$

ACUMULACIONES COMO PRODUCTORAS

ACUMULADOR - PARTE DE LA ESTRUCTURA

EN GENERAL:

$$
\begin{aligned}
prod &:: (a, b) \rightarrow b \\
cons &:: b \rightarrow c \\
fus &:: (a, c) \rightarrow c \\
cons\ (prod\ (t, z)) &= fus\ (t, cons\ z)
\end{aligned}
$$

ACUMULACIONES COMO PRODUCTORAS

APLICACIÓN SHORT-CUT FUSION (CONSUMIDORA)

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$\begin{aligned} horner\ [\,] &= 0 \\ horner\ (d : ds) &= d + 10 * horner\ ds \end{aligned}$$

$$\begin{aligned} horner &= fold\ (id, h) \\ &\textbf{where}\ \ h\ d\ r = d + 10 * r \end{aligned}$$

ACUMULACIONES COMO PRODUCTORAS

APLICACIÓN SHORT-CUT FUSION (PRODUCTORA)

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$areverse\ ([\,], xs) = xs$$
$$areverse\ (a : as, xs) = areverse\ (as, a : xs)$$

$$areverse = build\ grev$$

where

$$grev\ (fn, fc)\ ([\,], xs) = xs$$
$$grev\ (fn, fc)\ (a : as, xs) = grev\ (fn, fc)\ (as, fc\ \ a\ xs)$$

ACUMULACIONES COMO PRODUCTORAS

APLICACIÓN SHORT-CUT FUSION (PRODUCTORA)

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$areverse\ ([\,], xs) = xs$$
$$areverse\ (a : as, xs) = areverse\ (as, a : xs)$$

$$areverse = build\ grev$$

where

$$grev\ (fn, fc)\ (as, xs) = grev'(fn, fc)\ (as, id\ \ xs)$$
$$grev'(fn, fc)\ ([\,], xs) = xs$$
$$grev'(fn, fc)\ (a : as, xs) = grev'(fn, fc)\ (as, fc\ \ a\ xs)$$

ACUMULACIONES COMO PRODUCTORAS

APLICACIÓN SHORT-CUT FUSION (PRODUCTORA)

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$areverse\ ([\,], xs) = xs$$
$$areverse\ (a : as, xs) = areverse\ (as, a : xs)$$

$$areverse = build\ grev$$

where

$$grev\ (fn, fc)\ (as, xs) = grev'(fn, fc)\ (as, fold\ ([\,], (:))\ \ xs)$$
$$grev'(fn, fc)\ ([\,], xs) = xs$$
$$grev'(fn, fc)\ (a : as, xs) = grev'(fn, fc)\ (as, fc\ \ a\ xs)$$

ACUMULACIONES COMO PRODUCTORAS

APLICACIÓN SHORT-CUT FUSION (PRODUCTORA)

$$anumber\ (ds, xs) = horner\ (areverse\ (ds, xs))$$

$$areverse\ ([\,], xs) = xs$$
$$areverse\ (a : as, xs) = areverse\ (as, a : xs)$$

$$areverse = build\ grev$$

where

$$grev\ (fn, fc)\ (as, xs) = grev'(fn, fc)\ (as, fold\ (fn, fc)\ \ xs)$$
$$grev'(fn, fc)\ ([\,], xs) = xs$$
$$grev'(fn, fc)\ (a : as, xs) = grev'(fn, fc)\ (as, fc\ \ a\ xs)$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

Objetivo : Explicitar la existencia del acumulador.

$$build :: (\forall\, b\, .\, (b, a \to b \to b) \to c \to b) \to c \to [a]$$
$$build\ g = g\ ([\,], (:))$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

Objetivo : Explicitar la existencia del acumulador.

$$\begin{aligned} \mathit{buildA} &:: (\forall\, b\,.\,(b, a \to b \to b) \to (c, b) \to b) \to (c, [a]) \to [a] \\ \mathit{buildA}\ g &= g\ ([\,], (:)) \end{aligned}$$

Acumulaciones como productoras

Short-Cut Fusion Acumulativa

Objetivo : Explicitar la existencia del acumulador.

$$buildA :: (\forall\, b\, .\, (b, a \to b \to b) \to (c, b) \to b) \to (c, [a]) \to [a]$$
$$buildA\ g = g\ ([\,], (:))$$

Regla Fold/Build Acumulativa

$$g :: \forall\, b\, .\, (b, a \to b \to b) \to (c, b) \to b$$

$$\Rightarrow$$

$$fold\ (fn, fc) \circ buildA\ g = g\ (fn, fc) \circ (id \times (fold\ (fn, fc)))$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

$$
\begin{aligned}
&\mathit{areverse} = \mathit{buildA}\ \mathit{grev} \\
&\quad \textbf{where} \\
&\qquad \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ ([\,], z) = z \\
&\qquad \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ (a : \mathit{as}, z) = \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ (\mathit{as}, \mathit{fc}\ a\ z)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{horner} = \mathit{fold}\ (0, h) \\
&\qquad \textbf{where}\ \ h\ d\ r = d + 10 * r
\end{aligned}
$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

$$
\begin{aligned}
&areverse = buildA\ grev \\
&\quad \textbf{where} \\
&\qquad grev\ (fn, fc)\ ([\,], z) = z \\
&\qquad grev\ (fn, fc)\ (a : as, z) = grev\ (fn, fc)\ (as, fc\ a\ z)
\end{aligned}
$$

$$
\begin{aligned}
&horner = fold\ (0, h) \\
&\quad \textbf{where}\ h\ d\ r = d + 10 * r
\end{aligned}
$$

$$anumber\ (ds, xs) = grev\ (0, h)\ (ds, horner\ as)$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

$$
\begin{aligned}
&areverse = buildA\ grev \\
&\quad \textbf{where} \\
&\qquad grev\ (fn, fc)\ ([\,], z) = z \\
&\qquad grev\ (fn, fc)\ (a : as, z) = grev\ (fn, fc)\ (as, fc\ a\ z)
\end{aligned}
$$

$$
\begin{aligned}
&horner = fold\ (0, h) \\
&\quad \textbf{where}\ h\ d\ r = d + 10 * r
\end{aligned}
$$

$$anumber\ (ds, xs) = anumber'(ds, horner\ xs)$$

ACUMULACIONES COMO PRODUCTORAS

SHORT-CUT FUSION ACUMULATIVA

$$
\begin{aligned}
&\mathit{areverse} = \mathit{buildA}\ \mathit{grev} \\
&\quad \textbf{where} \\
&\qquad \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ ([\,], z) = z \\
&\qquad \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ (a : \mathit{as}, z) = \mathit{grev}\ (\mathit{fn}, \mathit{fc})\ (\mathit{as}, \mathit{fc}\ a\ z)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{horner} = \mathit{fold}\ (0, h) \\
&\quad \textbf{where}\ h\ d\ r = d + 10 * r
\end{aligned}
$$

$$
\begin{aligned}
\mathit{anumber}\ (\mathit{ds}, \mathit{xs}) &= \mathit{anumber}'(\mathit{ds}, \mathit{horner}\ \mathit{xs}) \\
\mathit{anumber}'\ ([\,], z) &= z \\
\mathit{anumber}'\ (a : \mathit{as}, z) &= \mathit{anumber}'\ (\mathit{as}, a + 10 * z)
\end{aligned}
$$

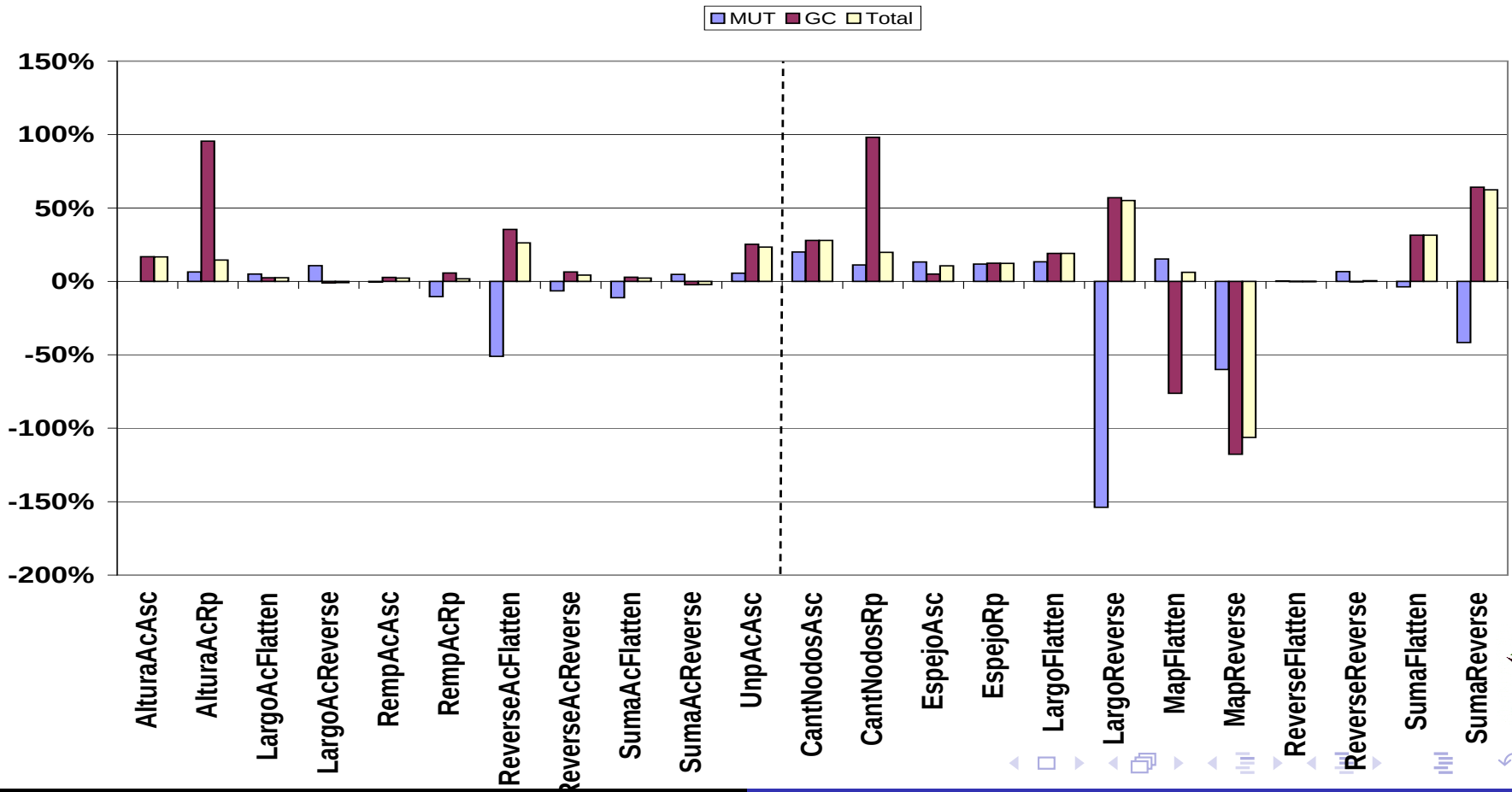
Pruebas realizadas

- Conjunto de programas simples
- Dos grupos según tipo consumidora
- Diferentes versiones (Original - Final)
- Medidas Tiempo / Espacio
- “Aislado” de las optimizaciones del compilador

Medidas de Tiempo

Relación Final/Original

$$\%Mejora = (T_{OR} - T_{FN})/T_{OR}$$



MapReverse - Códigos

mapReverse f $[x_1, x_2 \ \cdots \ , x_n]$ *ac*

Original

$$
\begin{aligned}
mapReverse\ f\ term\ ac &= map\ f\ (areverse\ (term, ac)) \\
map\ f\ [\,] &= [\,] \\
map\ f\ (x : xs) &= (f\ x) : (map\ f\ xs)
\end{aligned}
$$

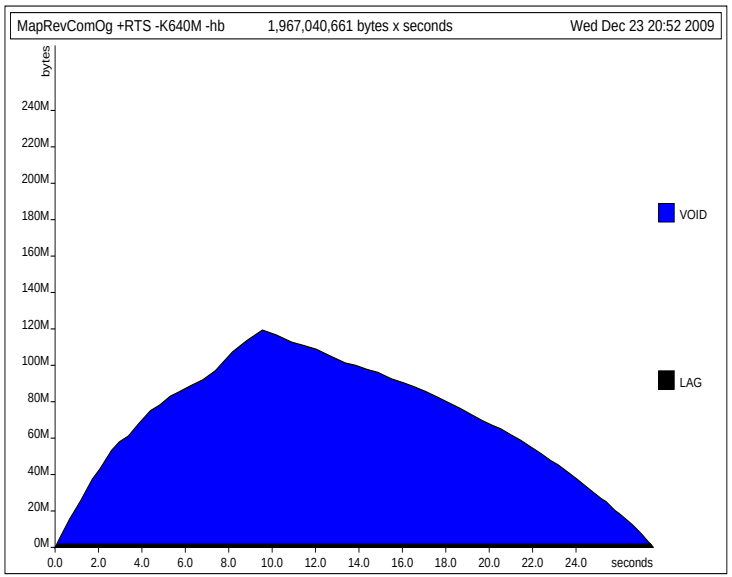
Final

$$
\begin{aligned}
mapReverse\ f\ term\ ac = &\ fus\ f\ (term, map\ f\ ac) \\
&\textbf{where} \\
&\quad fus\ f\ ([\,], ac) = ac \\
&\quad fus\ f\ (a : x, ac) = fus\ f\ (x, (f\ a) : ac)
\end{aligned}
$$

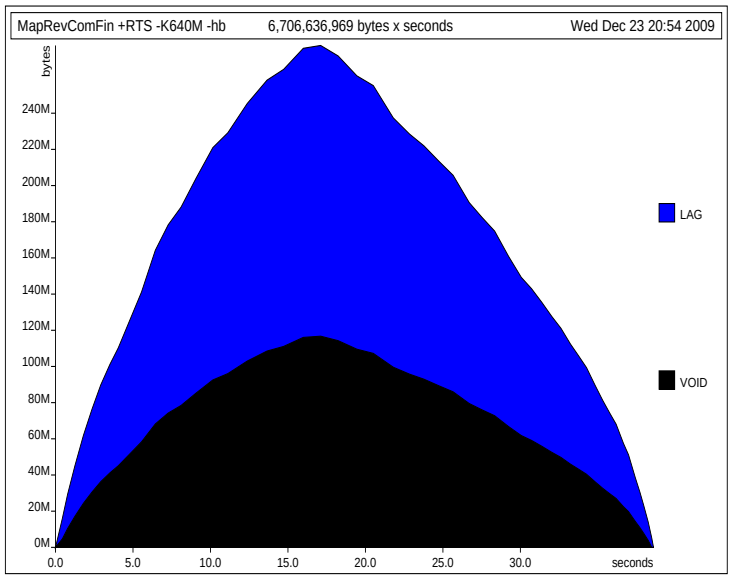
MapReverse - Reseña Biográfica

Tipos de celdas

Original



Final

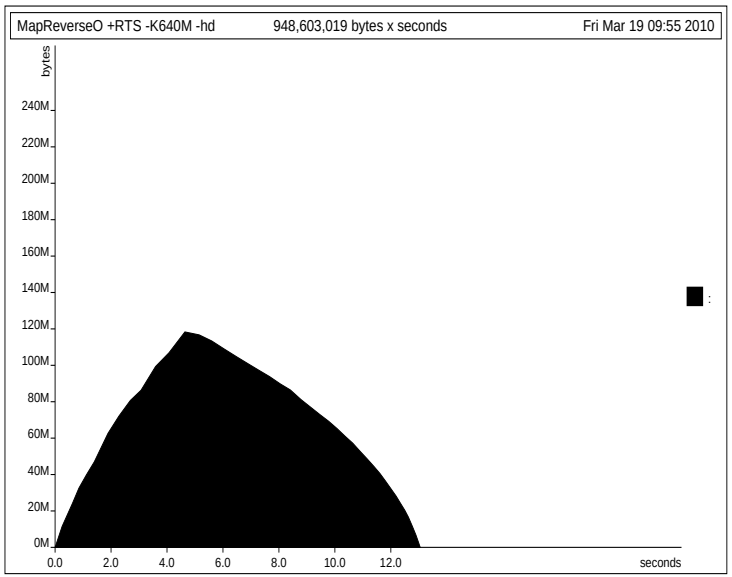


PEDECIBA

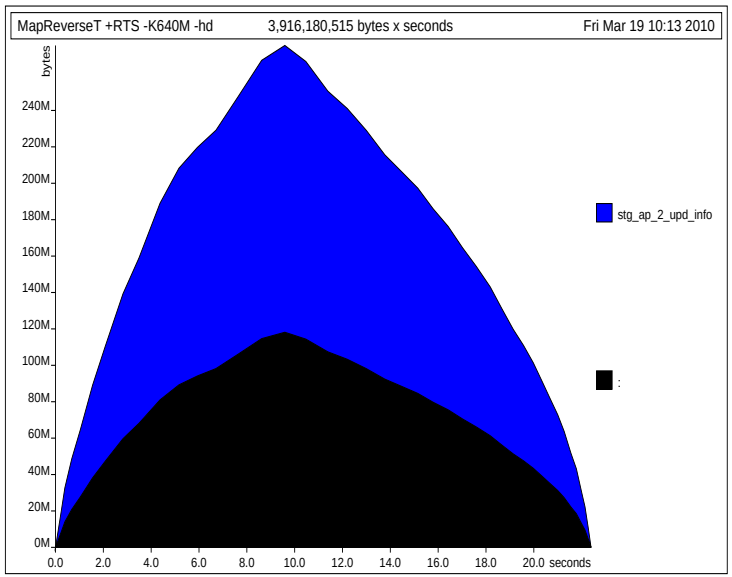
MapReverse - Reseña Histórica de Clausuras

Funciones creadoras

Original



Final



PEDECIBA

MapReverse - Evaluación

Original

$$map\ \ f\ \ (areverse\ ([x_1, x_2, \cdots, x_n], ac))$$

$$\ldots\ \ \ldots\ \ \ldots$$

$$= \quad f\ \ x_n : map\ \ f\ \ x_{n-1} : \cdots : x_2 : x_1 : ac$$

Final

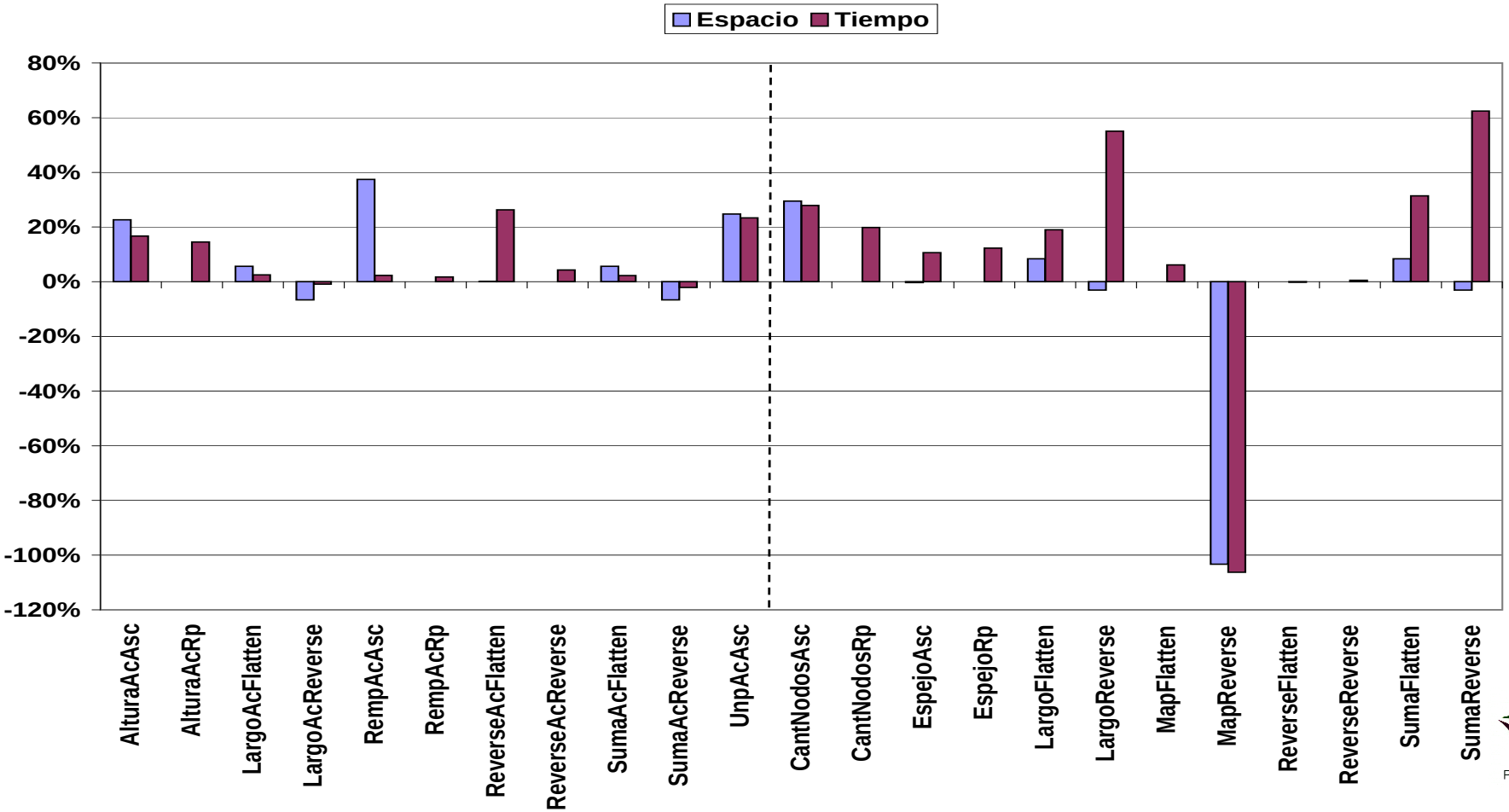
$$fus\ \ f\ \ ([x_1, x_2, \cdots, x_n], map\ \ f\ \ ac)$$

$$\ldots\ \ \ldots\ \ \ldots$$

$$= \quad f\ \ x_n : \cdots : f\ \ x_1 : map\ \ f\ \ ac$$

Tiempo - Espacio

Relación Final/Original



RESUMEN DE LAS PRUEBAS

- Programas considerados - Funciones productoras acumulativas

RESUMEN DE LAS PRUEBAS

- Programas considerados - Funciones productoras acumulativas
- Medidas de tiempo
  - Consumidoras Acumulativas $\longrightarrow$ Se mantiene el tiempo total
  - Consumidoras No acumulativas $\longrightarrow$ Mejora el tiempo total

RESUMEN DE LAS PRUEBAS

- Programas considerados - Funciones productoras acumulativas
- Medidas de tiempo
  - Consumidoras Acumulativas $\rightarrow$ Se mantiene el tiempo total
  - Consumidoras No acumulativas $\rightarrow$ Mejora el tiempo total
- Medidas de espacio $\rightarrow$ Mejoras en el espacio implican mejoras del tiempo

Contribución I

- Definición Ley Short-Cut Acumulativa
  - Explicita la existencia del acumulador en las productoras
  - Eficaz en la eliminación de la estructura de datos intermedia
  - Generalizable a cualquier tipo de datos
  - Pruebas Tiempo/Espacio

Contribución II

- Otro Enfoque: operador *afold*
  - Operador recursivo + leyes de fusión
  - Característica $\longrightarrow$ función de acumulación explícita
  - Menos restricciones $\longrightarrow$ Operador Extendido
    - Leyes de fusión
  - Caso de Estudio : Fisión

TRABAJOS RELACIONADOS

- Lazy Composition
  - Definiciones circulares
- Fusión Algebraica
  - Teoría de Monoides
- Derivación de Acumulaciones
  - Leyes de fusión
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion
- *dmap*
  - Operador *dmap* extensión al map

TRABAJOS RELACIONADOS

- Lazy Composition
  - Definiciones circulares $\downarrow$ Prod
- Fusión Algebraica
  - Teoría de Monoides
- Derivación de Acumulaciones
  - Leyes de fusión
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion
- *dmap*
  - Operador *dmap* extensión al map

Trabajos relacionados

- Lazy Composition
  - Definiciones circulares $\downarrow$ Prod
- Fusión Algebraica
  - Teoría de Monoides $\downarrow$ Prod
- Derivación de Acumulaciones
  - Leyes de fusión
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion
- *dmap*
  - Operador *dmap* extensión al map

PEDECIBA

TRABAJOS RELACIONADOS

- Lazy Composition
  - Definiciones circulares $\downarrow$ Prod
- Fusión Algebraica
  - Teoría de Monoides $\downarrow$ Prod
- Derivación de Acumulaciones
  - Leyes de fusión $\approx$ leyes de *afold*
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion
- *dmap*
  - Operador *dmap* extensión al map

TRABAJOS RELACIONADOS

- Lazy Composition
  - Definiciones circulares $\downarrow$ Prod
- Fusión Algebraica
  - Teoría de Monoides $\downarrow$ Prod
- Derivación de Acumulaciones
  - Leyes de fusión $\approx$ leyes de *afold*
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion Prod $\approx$ *zip* / Cons $=$ *foldl*
- *dmap*
  - Operador *dmap* extensión al map

TRABAJOS RELACIONADOS

- Lazy Composition
  - Definiciones circulares $\downarrow$ Prod
- Fusión Algebraica
  - Teoría de Monoides $\downarrow$ Prod
- Derivación de Acumulaciones
  - Leyes de fusión $\approx$ leyes de *afold*
- *destroy / unfold*
  - Regla dual a Short-Cut Fusion Prod $\approx$ *zip* / Cons $=$ *foldl*
- *dmap*
  - Operador *dmap* extensión al map Muy restrictivo

Trabajo Futuro

- Automatización
- Programas de mayor complejidad
- Optimizaciones al programa transformado
- Interacción de optimizaciones

Fusión en presencia de acumuladores

Mónica Martínez

Facultad de Ingeniería
Universidad de la República
Pedeciba Informática

Maestria en Informática
Orientador: Alberto Pardo