Trabajo Práctico 2

Santiago Bussanich
Ernesto Savio
Valentin Sosa

Junio 2025

1 Costos de la implementación de secuencias con listas

Si consideramos el costo del constructor : de listas constante, luego el trabajo de `mapS` para una secuencia $(x_0 : xs)$ de longitud n es

$$W_{mapS}(n) = W_f(x_0)+W_{mapS}(n-1) = W_f(x_0)+W_f(x_1)+\ldots+W_f(x_n) \in O(\sum_{i=0}^{n-1} W_f(x_i))$$

Para la profundidad, nuevamente considerando una secuencia $(x_0 : xs)$ de longitud n tenemos que

$$S_{mapS}(n) = max\{S_f(x_0), \max_{i=1}^{n-1} S_f(x_i)\} \in O(\max_{i=0}^{n-1} S_f(x_i))$$

Sea $n =$ `lengthS` s. Los dos primeros patrones de `reduceS` son casos base con trabajo y profundidad constante. Para $n \geq 2$ se llama a

$$\texttt{red}\big((1, x_1), \ldots, (1, x_n)\big)\ e\ [\,]\ op,$$

donde el mapeo inicial a pares $(1, x_i)$ cuesta $O(n)$.

Trabajo En `red` se consumen *dos* elementos de la lista en cada paso recursivo, se evalúa una sola operación *op* y, como mucho, se revisa el *stack* una vez mediante `eval`. Nuestra operación `red` no opera más de n elementos con f. De esta manera, en cada llamado recursivo de `eval` su trabajo está acotado y su coste total estará acotado por $\sum_{(x\oplus y)\in\mathcal{O}_r(\oplus,b,s)} W(x \oplus y)$.

$$W_{\texttt{red}}(n) \leq W_{\texttt{red}}(n-2) + O(1), \qquad \Longrightarrow \qquad W_{\texttt{red}}(n) = O(n).$$

El coste de `reduceS` está dominado por esta llamada:

$$W_{\texttt{reduceS}}(n) = O(n)\,.$$

Profundidad Cada invocación recursiva de `red` arranca *después* de que termine la anterior y `eval` tampoco se paraleliza, por lo tanto

$$S_{\texttt{red}}(n) = S_{\texttt{red}}(n-2) + O(1), \qquad \Longrightarrow \qquad S_{\texttt{red}}(n) = O(n).$$

Por lo tanto

$$S_{\texttt{reduceS}}(n) = O(n)\,.$$

Lo cual es esperable, ya que las listas no son paralelizables.
Finalmente, para `scanS` tenemos:

- Sea n la longitud de la secuencia de entrada s.
- En cada llamada recursiva, la función `contract` reduce el tamaño de la secuencia aproximadamente a $n/2$.
- El número de niveles de recursión es entonces $O(\log n)$, ya que en cada nivel el tamaño se reduce a la mitad.
- En cada nivel, las funciones `contract` y `expand` procesan todos los elementos de la secuencia de ese nivel, sumando un trabajo total de $O(n)$ a lo largo de todos los niveles.
- Si $\oplus$ no es constante, para el trabajo tenemos que sumar cada aplicación de $\oplus$ en cada nivel, esto es $\sum_{(x \oplus y) \in \mathcal{O}_s(\oplus, b, s)} W(x \oplus y)$. Por otro lado, como no estamos paralelizando las operaciones, es esperable que nuestra profundidad coincida con el trabajo.
- Entonces si $\oplus$ es constante, $W_{scanS}(n) \in O(n)$ y $S_{scanS}(n) \in O(n)$. Si $\oplus$ no es constante, tenemos que

$$W_{scanS}(n) = O(n + \sum_{(x \oplus y) \in \mathcal{O}_s(\oplus, b, s)} W(x \oplus y))$$

$$S_{scanS}(n) = O(n + \sum_{(x \oplus y) \in \mathcal{O}_s(\oplus, b, s)} S(x \oplus y))$$