Tipos Abstractos de Datos

Mauro Jaskelioff

20/04/2020

Colas

- ¿Qué es una cola?
- Es una estructura a la cual:
  - Podemos agregar elementos
  - Podemos obtener el primer elemento
  - Podemos quitar el primer elemento
  - Podemos preguntar si está vacía
  - Existe una relación entre el orden en que se agregan elementos y se sacan (FIFO).
- Esta descripción es *abstracta* porque refleja el comportamiento y no la implementación.

Tipos Abstractos de Datos

- La idea de un tipo abstracto de datos es abstraer detalles de implementación.
- Un usuario es alguien que simplemente usa la abstracción.
- El implementador provee un implementación que se ajusta al comportamiento esperado.
- El usuario sólo puede suponer el comportamiento descripto.
- Podemos ser más precisos sobre el comportamiento de las colas mediante una *especificación*.

TADs

Un TAD consiste de:

1. Un nombre de tipo. Ej . *Cola*
2. Operaciones.

   tad *Cola* $(A : Set)$ **where**
   import *Bool*
   $vacia \quad : Cola\ A$
   $poner \quad : A \to Cola\ A \to Cola\ A$
   $primero : Cola\ A \to A$
   $sacar \quad : Cola\ A \to Cola\ A$
   $esVacia : Cola\ A \to Bool$

3. Especificación del comportamiento
   - Especificación algebraica
     - Se describen operaciones y ecuaciones entre operaciones
   - Modelos
     - Se describen operaciones y cómo se interpretan en un modelo matemático

Especificaciones Algebraicas

- Especificación algebraica para colas

$$
\begin{aligned}
&\mathit{esVacia}\ \mathit{vacia} && = \mathit{True}\\
&\mathit{esVacia}\ (\mathit{poner}\ x\ q) && = \mathit{False}\\
&\mathit{primero}\ (\mathit{poner}\ x\ \mathit{vacia}) && = x\\
&\mathit{primero}\ (\mathit{poner}\ x\ (\mathit{poner}\ y\ q)) && = \mathit{primero}\ (\mathit{poner}\ y\ q)\\
&\mathit{sacar}\ (\mathit{poner}\ x\ \mathit{vacia}) && = \mathit{vacia}\\
&\mathit{sacar}\ (\mathit{poner}\ x\ (\mathit{poner}\ y\ q)) && = \mathit{poner}\ x\ (\mathit{sacar}\ (\mathit{poner}\ y\ q))
\end{aligned}
$$

- No confundir especificación con implementación!

Modelos

- Como modelo de colas tomamos las secuencias $\langle x_1, x_2, \ldots, x_n \rangle$
- Para cada operación damos una función equivalente sobre modelos:

$$
\begin{array}{lll}
\mathit{vacia} & = \langle\rangle & \\
\mathit{poner}\ x\ \langle x_1, x_2\ \ldots, x_n \rangle & = \langle x, x_1, x_2, \ldots, x_n \rangle & \\
\mathit{sacar}\ \langle x_1, x_2, \ldots, x_n \rangle & = \langle x_1, x_2, \ldots, x_{n-1} \rangle & \\
\mathit{primero}\ \langle x_1, x_2, \ldots, x_n \rangle & = x_n & \\
\mathit{esVacia}\ \langle x_1, x_2, \ldots, x_n \rangle & = \mathit{True} & \mathit{si}\ n = 0 \\
\mathit{esVacia}\ \langle x_1, x_2, \ldots, x_n \rangle & = \mathit{False} & \mathit{en\ otro\ caso}
\end{array}
$$

Implementaciones

- Cada TAD admite diferentes implementaciones
- Ejercicio:
  a) Implementar en Haskell el TAD de colas usando listas.

$$
\begin{array}{l}
\textbf{tad } \mathit{Cola}\ (A : \mathit{Set})\ \textbf{where} \\
\quad \begin{array}{ll}
\mathit{vacia} & : \mathit{Cola}\ A \\
\mathit{poner} & : A \to \mathit{Cola}\ A \to \mathit{Cola}\ A \\
\mathit{primero} & : \mathit{Cola}\ A \to A \\
\mathit{sacar} & : \mathit{Cola}\ A \to \mathit{Cola}\ A \\
\mathit{esVacia} & : \mathit{Cola}\ A \to \mathit{Bool}
\end{array}
\end{array}
$$

  b) ¿Cuál es el trabajo de las operaciones?

Costo de las implementaciones

- Hay dos posibles implementaciones con listas.
  1. Se agregan elementos al final, se sacan de la cabeza.
  2. Se agregan elementos a la cabeza, se sacan del final.
- Los dos métodos son lineales en alguna operación.
- ¿Cómo implementar una cola eficiente?

Otra implementación de Colas

- Implementemos colas usando un par de listas (xs, ys) tal que los elementos en orden sean $xs +\!\!+ reverse\ ys$
- Invariante de la implementación: Si xs es vacía, entonces ys también (las operaciones deben conservar este invariante...)

$$
\begin{aligned}
&\textbf{type } Cola\ a = ([a], [a]) \\
&vacia &&= ([\,], [\,]) \\
&poner\ x\ (ys, zs) &&= validar\ (ys, x : zs) \\
&prim\ (x : xs, ys) &&= x \\
&sacar\ (x : xs, ys) &&= validar\ (xs, ys) \\
&esvacia\ (xs, ys) &&= null\ xs \\
&validar\ (xs, ys) &&= \textbf{if } null\ xs \textbf{ then } (reverse\ ys, [\,]) \\
&&&\qquad\qquad \textbf{else } (xs, ys)
\end{aligned}
$$

Costo de esta implementación

La implementación con pares tiene los siguientes costos:

- $W_{vacia} \in O(1)$
- $W_{esVacia}(xs, ys) \in O(1)$
- $W_{poner}(xs, ys) \in O(1)$
- $W_{primero}(xs, ys) \in O(1)$
- $W_{sacar}(xs, ys) \in O(|ys|), O(1)_{(\text{amortizado})}$

Especificación de costo

- Cada TAD admite diferentes implementaciones
- En cada implementación las operaciones pueden tener diferentes costos (trabajo, profundidad).
- Dependiendo del *uso* de la estructura, puede convenir una implementación u otra.
- Por lo tanto es importante tener una *especificación de costo* de cada implementación.
  - Ejemplo: Para la 2da implementación de colas

$$W_{vacia} \in O(1) \qquad W_{poner}(xs, ys) \in O(1) \qquad W_{primero}(xs, ys) \in O(1$$
$$W_{sacar}(xs, ys) \in O(|ys|), O(1)_{(\text{amortizado})} \qquad W_{esVacia}(xs, ys) \in O(1$$

TADs en Haskell

- Una forma de implementar un TAD en Haskell es mediante una clase de tipos

```
class Cola t where
  vacia    :: t a
  poner    :: a → t a → t a
  sacar    :: t a → t a
  primero  :: t a → a
  esVacia  :: t a → Bool
```

- Una implementación es una instancia

```
instance Cola [] where
  vacia        = []
  poner x xs   = x : xs
  sacar xs     = init xs
  primero      = last
  esVacia xs   = null xs
```

Usando TADs en Haskell

- Usamos un TAD con una función polimórfica en el TAD

$$
\begin{aligned}
&\mathit{ciclar} && :: \mathit{Cola}\ t \Rightarrow \mathit{Int} \rightarrow t\ a \rightarrow t\ a \\
&\mathit{ciclar}\ 0\ \mathit{cola} && = \mathit{cola} \\
&\mathit{ciclar}\ n\ \mathit{cola} && = \mathit{ciclar}\ (n-1)\ (\mathit{poner}\ (\mathit{primero}\ \mathit{cola}) \\
& && \qquad\qquad (\mathit{sacar}\ \mathit{cola}))
\end{aligned}
$$

- Notar que la función *ciclar* funciona para cualquier instancia de Cola.
- La función *ciclar* no puede suponer nada acerca de la implementación.

TADs

- **Especificación:** Qué operaciones tiene el TAD y cómo se comportan. Es Única.
- **Implementación:** Cómo se realizan las operaciones y cuánto cuestan. Puede haber varias implementaciones (con diferentes costos). Todas deben garantizar el comportamiento dado por la especificación.
- **Uso:** Sólo puede suponer el comportamiento dado por la especificación. Se elije implementación de acuerdo al uso (menor costo para un determinado uso).

Conclusiones

Tipos Abstractos de Datos

- Ocultan detalles de implementación.
- El comportamiento se describe algebraicamente o proveyendo un modelo.
- Cada implementación debe tener una especificación de costo.