Justificación de la Corrección del Método Comerciar

Edgar Aguadé Nadal

May 2024

`void City::trade(City &visitor_city, const ProductSet &product_set);`

Precondición Cierto.

Postcondición Las ciudades han comerciado entre si todos aquellos productos posibles. O de un modo más formal:

Sean:

- $city_{local}$:= Parámetro implícito
- $id(product)$:= El identificador del producto "product"
- $exc(product)$:= Exceso del producto "product"

$\forall product_{local} \in city_{local} \forall product_{visitor} \in city_{visitor}$ tales que:
$id(product_{local}) = id(product_{visitor}) \wedge exc(product_{local}) * exc(product_{visitor}) < 0$

Se cumple que, siendo $t := min(|exc(product_{local})|, |exc(product_{visitor})|)$
$exc(product_{local}) > 0 \Longrightarrow own_{local} = own_{local} - t \wedge own_{visitor} = own_{visitor} + t$
$exc(product_{local}) < 0 \Longrightarrow own_{local} = own_{local} + t \wedge own_{visitor} = own_{visitor} - t$

O lo que viene a ser lo mismo, todos aquellos productos donde una ciudad tenga más de las que necesita, la otra menos de las que necesita y viceversa, han sido comerciados. La cantidad comerciada "t" es el mínimo de los excedentes, pues no pueden haber superávids y ni que una ciudad que tiene exceso quede con menos unidades de las que necesita.

Invariante

- $city_{local}.begin() \leq product_{local} \leq city_{local}.end()$
- $city_{visitor}.begin() \leq product_{visitor} \leq city_{visitor}.end()$
- $\forall product_i < product_{local} \in city_{local} \forall product_j < product_{visitor} \in city_{visitor}$ $city_{local}, city_{visitor}$ han intercamiado dichos productos cumpliendo la Postcondición

1 Justificación

1.1 Inicializaciones

Inicialmente no se ha comerciado ningún producto por lo que inicializamos ambos iteradores a al inicio de sus respectivos inventarios: $product_{local} = city_{local}.begin() \wedge product_{visitor} = city_{visitor}.begin()$, cumpliendo así las dos partes del *Invariante*, por un lado que ambos iteradores se encuentran entre el principio y el final, y, por otro lado, que todos los productos anteriores a los iteradores han sido comerciados.

1.2 Codición de salida

Se puede salir del bucle por dos razones:

- $product_{local} = city_{local}.end()$

La cual cosa indicaria, por el invariante, que hemos explorado todos los productos de $city_{local}$ y, en consecuencia, todos los de $city_{visitor}$ tales que $id(product_j) \leq product_{local}$.

Por lo tanto, $\forall product_j$ tal que $id(product_j) > product_{local}$ $\not\exists product_i$ tal que $id(product_i) = id(product_j)$

De lo contario significaria que $\exists product_j$ tal que $id(product_j) > product_{local} \wedge \exists product_i$ tal que $id(product_i) = id(product_j)$. Por el invariante obtenemos que $id(product_i) \geq city_{local}.end()$ lo cual es absurdo pues no hay mas productos despues del final del inventario.

- $product_{visior} = city_{visitor}.end()$

Completamente análogo a lo acabado de demostrar.

1.3 Cuerpo del bucle

Para analizar correctamente el cuerpo del bucle diferenciaremos claramente en tres casos:

- $id(product_{local}) < id(product_{visitor})$

Por el *Invariante* todos los productos anteriores a $product_{local}, product_{visitor}$ se han intercambiado si cumplian con $id(product_{local}) = id(product_{visitor})$. En estar ordenados de menor a mayor índice deducimos que $\not\exists product_j \in city_{visitor}$ tal que $id(product_j) < id(product_{visitor}) \wedge id(product_{local}) = id(product_j)$ De lo contrario deduciriamos que $\exists product_j \in city_{visitor}$ tal que $id(product_j) < id(product_{visitor})$ que no ha comerciado, absurdo por el *Invariante*. De este modo, avanzamos/descartamos $product_{local}$ puesto que estamos seguros que ese producto no va a ser comerciado.

- $id(product_{local}) > id(product_{visitor})$

$id(product_{local}) > id(product_{visitor}) \iff id(product_{visitor}) < id(product_{local})$ y por lo tanto, análogo al caso anterior.

- $id(product_{local}) = id(product_{visitor})$

Para que se vuelva a cumplir el *Invariante* las ciudades deben comerciar si se cumple $exc(product_{local}) * exc(product_{visitor}) < 0$. Con este objetivo precalculamos "t" la cantidad transacciónada. del modo que:

$$t := min(|exc(product_{local})|, |exc(product_{visitor})|)$$

De este modo podemos ver que actuamos según dos casos:

$exc(product_{local}) > 0 \implies exc(product_{visitor}) < 0$ por lo que $city_{local}$ vende y $city_{visitor}$ compra: $own_{local} = own_{local} - t \wedge own_{visitor} = own_{visitor} + t$

Por otro lado...

$exc(product_{local}) < 0 \implies exc(product_{visitor}) > 0$ por lo que $city_{local}$ compra y $city_{visitor}$ vende: $own_{local} = own_{local} + t \wedge own_{visitor} = own_{visitor} - t$

A su vez, en cada paso actualizamos los pesos y volumenes totales correspondientes de las ciudades.

Por último, avanzamos $product_{local}, product_{visitor}$ para cumplir con el *Invariante*.

1.4 Finalización

Como podemos apreciar, en cada iteración o avanzamos $product_{local}$ o avanzamos $product_{visitor}$ o ambos simultaneamente. Por lo que la distancia entre estos y $city_{local}.end(), city_{visitor}.end()$ respectivamente, disminuye en cada iteración de lo que deducimos que eventualmente el bucle finalizará.