Verificación de Propiedades de un Tipo Abstracto de Datos usando su Especificación Algebraica

Pablo Sánchez
Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria (Santander, España)
p.sanchez@unican.es

Resumen

El presente documento ilustra mediante un sencillo ejemplo cómo se pueden usar las especificaciones algebraicas para demostrar formalmente, es decir, de forma matemática, ciertas propiedades de un Tipo Abstracto de Datos. La demostración rigurosa de dichas propiedades es una condición necesaria para la construcción de software de alta fiabilidad cuya corrección tenga que ser verificada formalmente, certificando por tanto que dicho software es correcto y carece de errores.

1. Introducción

Cuando construimos software que usa *Tipos Abstractos de Datos (TADs)* es muy frecuente que nos surja la necesidad de razonar sobre ciertas propiedades de dichos TADs. Por ejemplo, una precondición de un método podría satisfacerse siempre y cuando, dada una *lista* o *secuencia* cualesquiera, la inversa de la inversa de dicha lista sea igual a dicha lista. O dicho de forma más algebraica, siempre y cuando la ecuación $invertir(invertir(l)) == l$ fuese verdadera para todo valor de l, siendo l una lista. Tal propiedad, que a primera vista parece obvia, podría no serlo. Si estamos construyendo software para un sistema crítico, como puede ser software para un avión o para el control del reactor de una central nuclear, necesitamos usar una base más sólida que nuestra mera intuición en aras de asegurar que el software está libre de fallos y por tanto podemos confiar en él.

Partiendo de una especificación algebraica, podemos demostrar en base a las ecuaciones de dicha especificación, y normalmente usando inducción, ciertas propiedades de un TAD. Una vez probadas matemáticamente dichas propiedades podemos tener la certeza absoluta de que son ciertas. Una vez demostrada una propiedad a nivel de especificación, toda implementación que sea conforme a dicha especificación poseerá automáticamente tal propiedad, y dicha propiedad se podrá usar para verificar que el software está libre de errores. Por ejemplo, se

puede usar para verificar que ninguna llamada a un método viola la precondición de dicho método.

A lo largo de este documento mostraremos como probar para una lista que la propiedad $invertir(invertir(l)) == l$ es verdadera. La Sección 2 proporciona la especificación algebraica del TAD Lista. La Sección 3 contiene la demostración por inducción de la propiedad. La Sección 4 contiene un breve sumario del documento.

2. Especificación Algebraica del TAD Lista

Esta sección contiene la especificación algebraica del TAD Lista, la cual se muestra en las Figuras 1, 2 y 3. Las ecuaciones de interés para este documento son las que afectan a la operación `addHead` (Figura 1, ec. 16 y 17) y a la operación `invertir` (Figura 3, ec. 44 y 45).

```
1  espec Sequence (List)
2
3  usa Boolean, Natural, Entero
4  generos Sequence(Element) como Seq
5  parametro
6       generos Element como E
7       operaciones
8            equal: Element Element -> Boolean
9       variables
10           x, y, z : Element;
11      ecuaciones
12           equal(x,x) = TRUE
13           equal(x,y) = equal(y,x)
14           [equal(x,y) and equal(y,z)] equal(x,z) = TRUE;
15 fparametro
16 operaciones
17      emptySeq  : -> Seq
18      addTail, addHead : Seq E -> Seq
19      add       : Seq E Natural -> Seq
20      getTail, getHead : Seq -> E
21      get       : Seq Natural -> E
22      replace   : Seq Natural E -> Seq
23      removeAll, removeFirst, removeLast : Seq E -> Seq
24      remove    : Seq Natural -> Seq
25      contains  : Seq E -> Boolean
26      findFirstIndex, findLastIndex : Seq E -> Entero
27      size      : Seq   -> Natural
28      number    : Seq E -> Natural
29      isEmpty   : Seq   -> Boolean
30      concat, equal : Seq Seq -> Seq
31      equal         : Seq Seq -> Boolean
32      invertir  : Seq -> Seq
33 fspec (continua)
```

Figura 1: Especificación Algebraica del TAD Lista (0)

```
1  espec Sequence (List) (continuación)
2
3  vars
4       s, s2 : Seq; n : Natural; x, y : Element;
5
6  precondiciones
7       [0 <= n <= size(s)] add(s,n)
8       [isEmpty(s) == FALSE] getTail(s), getHead(s)
9       [0 <= n < size(s)] get(s,n), remove(s,n), replaces(s,n,x)
10
11 ecuaciones
12       // Las generadoras son emptySeq y addTail y son libres
13       // Los patrones del tipo son:
14       //   - emptySeq
15       //   - addTail(s,x)
16       addHead(emptySeq,x)  = addTail(emptySeq,x)
17       addHead(addTail(s,x),y) = addTail(addHead(s,y),x)
18
19       size(emptySeq) = 0
20       size(addTail(s,x)) = 1 + size(s)
21
22       add(emptySeq,x,0) = addTail(emptySeq,x)
23       [n == size(addTail(s,x))]
24           add(addTail(s,x),y,n) = addTail(addTail(s,x),y)
25       [n < size(addTail(s,x))]
26           add(addTail(s,x),y,n) = addTail(add(s,y,n),x)
27
28       getTail(addTail(s,x)) = x
29
30       getHead(addTail(emptySeq,x))  = x
31       getHead(addTail(addTail(s,x))) = getHead(addTail(s,x))
32
33       [n == (size(s) - 1)] get(addTail(s,x),n) = x
34       [n  < (size(s) - 1)] get(addTail(s,x),n) = get(s,n-1)
35
36            removeAll(emptySeq,x) = emptySeq
37       [equal(x,y) == TRUE]
38           removeAll(addTail(s,x),y) = removeAll(s,y)
39       [equal(x,y) != TRUE]
40           removeAll(addTail(s,x),y) = addTail(removeAll(s,y),x)
41
42       [n == (size(s) - 1)]
43           remove(addTail(s,x),n) = s
44       [n <  (size(s) - 1)]
45           remove(addTail(s,x),n) = addTail(remove(s,n),x)
46
47       replace(s,n,y) = add(remove(s,n),y,n)
48
49       contains(emptySeq,x) = FALSE
50       [equal(x,y) == TRUE]
51           contains(addTail(s,x),y) = TRUE
52       [equal(x,y) != TRUE]
53           contains(addTail(s,x),y) = contains(s,y)
54 fspec (continua)
```

Figura 2: Especificación Algebraica del TAD Lista (1)

```
1  espec Sequence (List) (continuación)
2
3        findLastIndex(emptySeq,x) = -1
4        [(equal(x,y) == TRUE)]
5              findLastIndex(addTail(s,x),y) = size(s);
6        [(equal(x,y) != TRUE)]
7              findLastIndex(addTail(s,x),y) = findLastIndex(s,y);
8
9        findFirstIndex(emptySeq,x) = -1
10       [(equal(x,y) != TRUE)]
11             findFirstIndex(addTail(s,x),y) = findFirstIndex(s,y);
12       [(equal(x,y) == TRUE) and (contains(s,y) == TRUE)]
13             findFirstIndex(addTail(s,x),y) = findFirstIndex(s,y);
14       [(equal(x,y) == TRUE) and (contains(s,y) != TRUE)]
15             findFirstIndex(addTail(s,x),y) = size(s);
16
17       [findFirstIndex(s,x) != -1]
18           removeFirst(s,x) = remove(s,findFirstIndex(s,x));
19       [findFirstIndex(s,x) == -1] removeFirst(s,x) = s;
20
21       [findLastIndex(s,x) != -1]
22           removeLast(s,x) = remove(s,findLastIndex(s,x));
23       [findLastIndex(s,x) == -1]  removeLast(s,x) = s;
24
25       number(emptySeq,x) = 0
26       [equal(x,y) == TRUE]
27           number(addTail(s,x),y) = 1 + number(s,y)
28       [equal(x,y) != TRUE]
29           number(addTail(s,x),y) = number(s,y)
30
31       isEmpty(emptySeq)      = TRUE
32       isEmpty(addTail(s,x)) = FALSE
33
34       concat(s,emptySeq) = s
35       concat(s,addTail(s2,x))  = addTail(concat(s,s2),x)
36
37       equal(emptySeq,emptySeq)      = TRUE
38       equal(addTail(s,x),emptySeq) = FALSE
39       equal(emptySeq,addTail(s,x)) = FALSE
40       [equal(x,y) == TRUE]
41       equal(addTail(s,x),addTail(s2,y)) = equal(s,s2)
42       [equal(x,y) != TRUE] equal(addTail(s,x),addTail(s2,y)) =
             FALSE
43
44       invertir(emptySeq) = emptySeq
45       invertir(addTail(s,x)) = addHead(invertir(s),x)
46 fespec
```

Figura 3: Especificación Algebraica del TAD Lista (2)

3. Demostración por Inducción

En esta sección demostraremos la propiedad enunciada en la introducción de este documento, y que se muestra en la ec. (1). Usaremos para ello inducción matemática. Por tanto, debemos demostrar que dicha propiedad se cumple para un caso base, que será la secuencia vacía; asumiremos que es verdad para listas de tamaño n (hipótesis de inducción); y lo demostraremos para listas de tamaño $n+1$ usando para ello la hipótesis de inducción.

$$invertir(invertir(l, x)) = l \tag{1}$$

Caso Base Tenemos que demostrar la ecuación (1) es verdadera para la lista vacía, tal como muestra la ecuación (2).

$$invertir(invertir(emptySeq)) == emptySeq \tag{2}$$

Para ello reducimos el término izquierdo de la ecuación (2), comprobando que al reducirse se iguala al término derecho de dicha ecuación. Dicha reducción se muestra en (3).

$$\begin{aligned} &invertir(invertir(emptySeq)) = \\ &\qquad [Aplica\ Figura\ 3, ec.\ 44] = invertir(emptySeq) \\ &\qquad [Aplica\ Figura\ 3, ec.\ 44] = emptySeq \end{aligned} \tag{3}$$

Hipótesis de Inducción Asumimos, de acuerdo con la técnica de *demostración por inducción*, que la ecuación (4) es verdadera para lista de tamaño n.

$$invertir(invertir(l)) = l \tag{4}$$

Paso Inductivo Demostramos ahora, apoyándonos en la hipótesis de inducción, que la ecuación (1) es verdadera para lista de tamaño $n+1$. Es decir, debemos igualar los términos de la ecuación (5).

$$invertir(invertir(addTail(l, x))) = addTail(l, x) \tag{5}$$

Para ello tendremos que hacer uso de una nueva propiedad, que parece razonable, pero que hemos también de demostrar matemáticamente.

En (7) se muestra la reducción llevada a cabo para igualar el término izquierdo de la ecuación (11) con el término derecho de la misma. Para realizar el segundo paso de dicha reducción hemos asumido que la propiedad de la ecuación (6) es verdadera. Dicha propiedad dice que la inversa de una lista cuyo elemento en la cabeza es x es igual a la inversa de la lista sin la cabeza, y con dicha cabeza x como cola.

$$invertir(addHead(l, x)) = addTail(invertir(l), x) \tag{6}$$

$$
\begin{aligned}
&invertir(invertir(addTail(l,x))) = \\
&\qquad [Figura\ 3, ec.\ 45] = invertir(addHead(invertir(l),x)) \\
&\qquad [Prop.\ a\ demostrar] = addTail(invertir(invertir(l)),x) \\
&\qquad [Hip.\ inductiva] = addTail(l,x)
\end{aligned} \tag{7}
$$

La reducción (7) se basa en la propiedad de la ecuación (6). Por tanto, para que dicha reducción sea correcta debemos demostrar que tal propiedad es correcta. Para ello procedemos nuevamente por inducción sobre el tamaño de la lista.

Caso Base Hemos de demostrar que la propiedad es cierta para listas vacías, tal como muestra la ecuación (8).

$$
invertir(addHead(emptySeq,x)) = addTail(invertir(emptySeq),x) \tag{8}
$$

Para ello reducimos los términos derechos e izquierdo hasta llegar a un mismo término, tal como muestran las ecuaciones (**??**) y (10).

$$
\begin{aligned}
&invertir(addHead(emptySeq,x)) = \\
&\qquad [Aplica\ Figura\ 2, ec.\ 16] = invertir(addTail(emptySeq,x)) \\
&\qquad [Aplica\ Figura\ 3, ec.\ 45] = addHead(invertir(emptySeq),x) \\
&\qquad [Aplica\ Figura\ 3, ec.\ 44] = addHead(emptySeq,x) \\
&\qquad [Aplica\ Figura\ 2, ec.\ 16] = addTail(emptySeq,x)
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
&addTail(invertir(emptySeq),x) = \\
&\qquad [Aplica\ Figura\ 3, ec44] = addTail(emptySeq,x)
\end{aligned} \tag{10}
$$

Hipótesis de Inducción Asumimos como verdadera la ecuación (6) para listas de tamaño n.

Paso Inductivo Demostramos que la ecuación (11) es verdadera haciendo uso de la hipótesis de inducción. Para ello reducimos el término izquierdo de la ecuación (11) para igualarlo con el término derecho de la misma. La ecuación (12) muestra dicha reducción.

$$
invertir(addHead(addTail(l,y),x)) = addTail(invertir(addTail(l,y)),x) \tag{11}
$$

$$
\begin{aligned}
&invertir(addHead(addTail(l,y),x)) = \\
&\qquad [Aplica\ Figura\ 1, ec.\ 17] = invertir(addTail(addHead(l,x),y)) \\
&\qquad [Aplica\ Figura\ 3, ec.\ 45] = addHead(invertir(addHead(l,x)),y) \\
&\qquad [Aplica\ Hip.\ Inductiva] = addHead(addTail(invertir(l),x),y) \\
&\qquad [Aplica\ Figura\ 1, ec.\ 17] = addTail(addHead(invertir(l),y),x) \\
&\qquad [Aplica\ Figura\ 3, ec.\ 45] = addTail(invertir(addTail(l,y)),x)
\end{aligned} \tag{12}
$$

Por tanto, demostrada la propiedad principal y la auxiliar que era necesaria para demostrar la principal, podemos concluir sin lugar a dudas que la inversa de la inversa de una lista es la propia lista. Dicha propiedad está demostrada matemática y será verdad para toda implementación que satisfaga o se corresponda con la especificación usada para demostrar la propiedad.

4. Sumario

Este documento ha presentado un sencillo ejemplo que ilustra cómo se pueden utilizar las especificaciones algebraicas para demostrar mediante inducción matemática ciertas propiedades de un Tipo Abstracto de Datos. Demostrar dichas propiedades puede resultar crucial de cara a la elaboración de software fiable que necesitemos verificar que se encuentra libre de errores. Por desgracia, estos procedimientos de verificación requieren tiempo y cierta habilidad, lo que los hace bastante costosos. Por tanto, se suelen emplear exclusivamente en sistemas software críticos, donde un mínimo fallo puede acarrear consecuencias funestas.