Sobre especificaciones y modelos

C. Zoltan de Torres
Departamento de Computación y Tecnologí de la Información
Universidad Simón Bolívar
Apartado 89000
Caracas 1080
Venezuela

Marzo 1991

Resumen

Se analiza usando un ejemplo, que la calidad de las especificaciones depende fuertemente del modelo utilizado para describirlas. En particular cada modelo de solución puede aportar propiedades a la especificación, que dificilmente fueran deducibles de la especificación informal. Se presentan cuatro soluciones de un mismo problema, cada una de ellas con características muy distintas desde el punto de vista computacional. La comparación de estas soluciones es difícil sin tener en cuenta la arquitectura donde corra la aplicación, sin embargo cada una de ellas aporta a la especificación información del problema que contribuye a la reusabilidad de las implementaciones desarrolladas, y al test de implementaciones futuras.

Abstract

We show, by means of an example, that the quality of specifications depends strongly on the model used to describe them. In particular, each such model can suggest properties for the formal specification which would be hard to to deduce from an informal description. We present four solutions to the same problem, each with very different characteristics from the computational point of view. The relative efficiency of these solutions is strongly dependent on the underlying

system architecture, nevertheless they provide information that can be useful for software reuse and for testing future developements.

1 Metodología de las especificaciones

El proceso usual de desarrollo de software, comienza con una especificación informal del problema a resolver. La continuación natural de este proceso, para los defensores de las especificaciones formales, es la formalización de la descripción informal.

Existe una variedad muy grande de lenguajes de especificación, y dentro de cada familia existen miembros que se diferencian solamente en la semántica elegida. Algunos lenguajes algebráicos seleccionan ciertos modelos particulares: álgebra inicial, final, finitamente generada. Veremos, usando un mismo ejemplo, que ciertas soluciones computacionalmente interesante se dejan fuera de los modelos de la especificación. Por el otro lado, las soluciones encontradas nos proveen de una mejor caracterización del problema y permite tener una visión de familias de soluciones posibles basadas en propiedades de modelos particulares.

Este trabajo parte de una especificación informal del problema a resolver, elaborando una solución computacional del mismo.

Esta solución nos permite asegurar la existencia de al menos una solución computacionalmente aceptable para el problema. Asimismo la solución nos sirve de punto de partida para tipificar que es lo escencial y que es prescindible en las soluciones del problema dado.

2 El problema

El Problema: especificación informal [Extraído de "Concrete Mathematics"] [KNU2].

El problema de nuestro interés es un problema muy antiguo y conocido por el nombre de un historiador del siglo I: Flavious Josephus.

Durante la guerra Judeo-Romana, Josephus se encuentra formando parte de un grupo de 41 rebeldes. Atrapados en una cueva, por los Romanos, prefiriendo el suicidio a la rendición, los rebeldes deciden formar un círculo y 'suicidar' al tercero en la cuenta hasta que todos los soldados fueran eliminados.

Josephus, contrario al suicidio, hace rápidamente las cuentas de manera de colocar a su amigo y a él mismo en posiciones apropiadas del círculo, de manera que resultasen los últimos elegidos en este proceso de suicidio. Gracias a su habilidad matemática y a su velocidad de cálculo es que Josephus pudo contar la historia.

Nosotros simplificaremos el problema, buscando un mecanísmo rápido de cálculo. Cambiaremos la forma de designar al próximo suicida siendo 2 y no 3 como en el problema original. Asimismo, consideraremos que puede quedar un solo sobreviviente en el círculo después de la operación de suicidio.

El Problema: Especificación formal

Nuestro problema se ha reducido a la especificación de una función J_2. Usaremos para la presentación la sintaxis de PLUSS [GAU].

Se define un nuevo tipo llamado JOSEPHUS, que utiliza como tipos primitivos los tipos EJERCITO (que corresponde a la organización de un conjunto de soldados con un operador total sucesor) y booleano.

spec :	JOSEPHUS		
use:	ejército,bool		
sort:	sobrevivientes		
generated by:			
operations:			
J_2 :	ejército	$\rightarrow$	sobrevivientes
vacío:		$\rightarrow$	ejército
member:	sobrevivientes ejército	$\rightarrow$	Bool
axioms:			
	soldados $\neq$ vacío $\Rightarrow$		
	$member(J_2(soldados), soldados)$	$= true$	
where:	*soldados* :	ejército	
end	JOSEPHUS		

Dado nuestro desconocimiento de la función, definimos como un nuevo **sort** el rango de la misma. Veremos que en la medida que nos familiarizamos con la función estaremos en mejores condiciones de caracterizar este conjunto

de valores. Esta caracterización nos permitirá establecer el predicado de identificación de este subconjunto (la función característica.)

Los axiomas, que son usados para describir las propiedades de las funciones, van a ser pocos, debido a nuestro desconocimiento del comportamiento de la función.

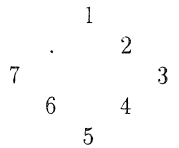
Indicamos en el único axioma de la presentación, que el rango de la función J_2 es un elemento del conjunto inicial de soldados atrapados en la cueva. A continuación veremos diferentes soluciones del problema. Durante el proceso de elaboración de las diferentes soluciones, usando modelos particulares, estaremos en condiciones de completar la especificación de la función. Esto nos permite establecer una jerarquía de especificaciones, mediante la noción de agregar axiomas, usualmente llamado enriquecimiento, operación esta que reduce la cantidad de modelos de la especificación.

La especificación raíz va siendo completada con la alimentación de las especificaciones hoja. A continuación damos las diferentes soluciones al problema, indicando en la subsección 'propiedades derivables de la solución', las propiedades generales de la función que se identifican en la elaboración de las soluciones particulares

SOLUCION 1

Dada la especificación formal la función J_2 esta subespecificada respecto al enunciado del problema, ya que según la especificación, cualquier soldado del conjunto de soldados iniciales puede resultar el sobreviviente (rango de J_2). Una forma de interpretar la especificación es que esta describe la elección de un elemento en un conjunto. Debemos agregar en la especificación la estrategia de eliminación decidida por los rebeldes. A fin de visualizar esa estrategia hacemos una inmersión de los soldados en un intervalo de los naturales. Esta inmersión se limita a asociar los soldados del conjunto con los puestos que ocupan en el círculo. A lo largo de esta solución hablaremos de J_2 como una función cuyo dominio debe interpretarse como el número de puestos en el círculo inicial y cuyo rango corresponde al puesto que ocupa el sobreviviente en el círculo inicial.

Dibujando el círculo y comenzando el proceso de eliminación (no teniendo en cuenta el número total de puestos en el círculo).



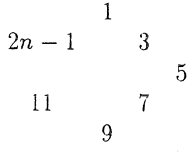
Se elimina cada 2 soldados y si el primero se salva (por el momento), el segundo es eliminado, el tercero se salva y se elimina el cuarto ...

Con este modelo podemos facilmente identificar propiedades del puesto del sobreviviente: es un puesto impar, dado que los puestos pares caen bajo la orden de suicidio en la primera vuelta.

Las órdenes de suicidio de la segunda vuelta dependerán de la paridad del tamaño del círculo inicial. Es de hacer notar que esta caracterización de los soldados suicidados en la primera vuelta no sería posible si hubieramos trabajado con los identificadores de los soldados ya que es un conjunto sin estructura.

Consideraremos ahora las diferentes posibilidades para n el número inicial de soldados.

Para el caso $n = 2m$, y considerando el círculo después de la primera ronda, tendría el siguiente aspecto:



después de haber eliminado al soldado en el puesto $2m$. Este soldado se elimina ya que es par.

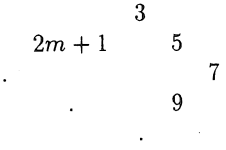
Observando el círculo resultado, podemos considerar que estamos ante la resolución del mismo problema pero con los nombres de los puestos cambiados. Renombrando los puestos y resolviendo el problema, nos resta construir la traducción efectiva de los nuevos puestos a la identificación original.

El nuevo tercer puesto corresponde al quinto puesto del círculo original.

La función inversa de reasignar nombres a los puestos, nos permite establecer la siguiente igualdad:

$$J_2(2m) = 2 * J_2(m) - 1 \tag{1}$$

Para el caso en que el número total de soldados sea impar $n = 2m + 1$: resulta que el soldado con que se inicia la cuenta (soldado 1) es eliminado después del soldado $2m$, resultando que al fin de la primera vuelta mas la eliminación del soldado del puesto 1 queda una configuración que trataremos de hacer corresponder a una configuración inicial:



Si este círculo lo consideramos como una configuración inicial con cambio de nombres. La conversión de puestos resulta en la siguente ecuación:

$$J_2(2m + 1) = 2 * J_2(m) + 1 \tag{2}$$

Las dos relaciones de recurrencia nos permiten establecer un sistema de ecuaciones:

AXIOMA 1	$J_2(2m)$	$=$	$2 * J_2(m) - 1$	$m \geq 1$
AXIOMA 2	$J_2(2m + 1)$	$=$	$2 * J_2(m) + 1$	$m \geq 1$
AXIOMA 3	$J_2(1)$	$=$	1	

Para implementar esta función se deberá considerar las alternativas posibles para el dominio de la función:

1. Los argumentos que se reciben pertenecen a valores posibles.

2. Se debe completar la función para los valores para los cuales no está definida.

Si el tipo del dominio es **cardinal**, debe completarse en 0, y una posible completación es que $J_2(0)$ coincida con la identidad:

$$J_2(0) = 0 \tag{3}$$

Si el tipo de argumento es **entero**, deberemos completar convenientemente para los valores negativos. Hacemos notar que esta completación hace que las álgebras (modelos de la especificación) cambien.

A continuación presentaremos una **implementación** considerando que el dominio de J_2 es cardinal. Supondremos que se dispone de una operación de shift a la derecha. Un posible modelo es 'lista' de booleanos con la interpretación:

$$shiftd(\sum_{0}^{\infty} a_i * 2^i) = \sum_{0}^{\infty} a_{i+1} * 2^i \tag{4}$$

La implementación estará anotada con precondiciones y poscondiciones.

IMPLEMENTACION

PRECONDICION $n \geq 0$
$J_2(n)$
caso n
0 : $J_2 := 0$;
1 : $J_2 := 1$;
sino : **PRECONDICION** $n \geq 2$ por lo que
$shiftd(n) \geq 1$
$A = 2 * J_2(shiftd(n))$
POSCONDICION $A \geq 2, par(A) = true$
si n es *par* entonces $J_2 := A - 1$
sino $J_2 := A + 1$
finsino
POSCONDICION $J_2 \geq 1, impar(J_2) = true$
End J_2

2.1 Propiedades derivables de la solución 1

Hacemos notar que esta implementación depende de un predicado **par**. Este predicado tiene una complejidad que es altamente dependiente del modelo de enteros que se utilice.

Una ley derivable en este momento es:

$$par(J_2(n)) = falso \tag{5}$$

Lo anterior debe inducir a pensar que incluyendo la ecuación anterior en la especificación, siempre se debiera implementar el predicado **par**. Sin embargo la ecuación anterior admite otra presentación:

$$\exists\, i \quad J_2(n) = suc^{2*i+1}(0) \tag{6}$$

La presentación utiliza la función sucesor sobre los naturales. Esta presentación sugiere los modelos de los enteros módulo p para algún p mayor que la talla del ejército. En ese caso la ecuación tiene infinitas soluciones.

SOLUCION 2

Dado que el sistema de ecuaciones recurrentes tiene una expresión cerrada, vamos a hacer el análisis de una implementación usando la expresión cerrada.

IMPLEMENTACION 2

Analizando la implementación 1 vemos que para todo valor de entrada ≥ 1 la operación de $shiftd(n) \geq 1$ y por lo tanto alcanza el valor 1. Es por eso que analizaremos las ecuaciones recurrentes tomandolas de a pares, considerando los casos en que solamente usan los axiomas 1 y 3 y luego los que utilizan los axiomas 2 y 3.

- **Axioma 1 y 3**

  El caso en consideración corresponde a argumentos potencia de 2.

  $J_2(2^r) = 2*J_2(2^{r-1})-1 = 2*(2*J_2(2^{r-2})-1)-1 = 2^2*J_2(2^{r-2})-2-1 = 2^2*(2*J_2(2^{r-3}-1)-2-1 = 2^3*J_2(2^{r-3})-2^2-2^1-2^0$

  y en forma general:

$J_2(2^r) = 2^i * J_2(2^{r-1}) - \Sigma_{j=0}^{i-1} 2^j \qquad i \leq r$

y para el caso $r = i$ resulta

$J_2(2^r) = 2^r * J_2(1) - \Sigma_0^{r-1} 2^j = 2^r - (2^r - 1) = 1$

Los cálculos nos han permitido identificar la propiedad:

$$J_2(2^r) = 1 \tag{7}$$

- **Axioma 2 y 3**

  El formato de los argumentos que solo utilizan los axiomas 2 y 3 es $n = 2^r - 1$

  $J_2(2^r - 1) = J_2(2 * (2^{r-1} - 1) + 1)$

  y usando el axioma 2 resulta

  $J_2(2^r - 1) = 2^i * J_2(2^{r-i} - 1) + \sum_0^{i-1} 2^j \qquad r > i$

  y tomando $i = r - 1$ resulta:

  $J_2(2^r - 1) = 2^r - 1$

Ahora estamos en condiciones de analizar el caso general, resultando

$$J_2(2^r + s) = 2 * s + 1,\ r \geq 0,\ 0 \leq s < 2^r \tag{8}$$

2.2 Propiedades derivables de la solución 2

Vamos a mostrar que este análisis de la implementación de la función de Josephus, nos permite agregar otra ecuación a la especificación. No eran propiedades evidentes obtenibles de la especificación informal de la función.

De la ecuación 8 podemos agregar a la especificación de la función de Josephus:

$$(J_2(n) = 1) = (\exists m \quad n = 2^m) \tag{9}$$

Esta presentación sugiere que una implementación de la función de Josephus pudiera ser utilizada como implementación del predicado que caracteriza a las potencias de 2. Asimismo la ecuación 8 puede ser útil para test de implementaciones que numeren los puestos de 1 a n.

Una pregunta inmediata, es si podemos independizar la función de Josephus de la representación de los enteros. Usaremos la maqueta física de imaginar los soldados en un círculo (esta solución utiliza el modelo sugerido por la especificación informal), e iremos eliminandolos del círculo.

Solución 3

En esta solución retendremos la inmersión de los soldados en el intervalo $[0 \ldots 1]$ y simulamos la operación de suicidio. Por lo tanto nuestro círculo estará formado por soldados vivos y soldados ya suicidados.

La designación de un soldado como el próximo al suicidarse se reduce a encontrar un soldado vivo y suicidar a su vecino (en una dirección determinada) que aún esté vivo.

IMPLEMENTACION

$J_2(n)$
si $n \leq 1$ **entonces** $J_2 = n$

PRECONDICION todos los soldados del grupo estan vivos y comenzará la cuenta con el soldado 1, que está vivo.

	sino	$s := 1; muertos := 0;$
0		**mientras** queden por eliminar
1		$s := s \bmod n + 1$
2		mientras s está muerto $s := s \bmod n + 1$
3		eliminar s; $muertos = muertos + 1$
4		mientras s está muerto $s := s \bmod n + 1$
		finmientras
	$J_2 := s$	

Al ciclo de la línea 0 se ingresa siendo s un soldado vivo, su vecino se obtiene en la línea 1. Este vecino debe ser un soldado vivo (línea 2). En la línea 3 se elimina al vecino vivo y se contabiliza el número de soldados suicidados. La línea 4 recupera la precondición requerida para el ciclo que se inicia en la línea 0. La condición de salida del ciclo se reduce a que el numero de muertos sea uno menos que el total original de soldados.

2.3 Propiedades derivables de la solución 3

La solución presentada esta parametrizada por los enteros módulo el tamaño del grupo inicial de soldados. Este tipo de presentaciones no se admite en un gran número de lenguajes de especificación.

Nuestra especificación admite como modelos álgebras finitas (por ejemplo enteros módulo p con $p > n$). Es de hacer notar que el predicado **par** (utilizado en la solución 1) no tiene sentido utilizando los enteros módulo p con p impar.

Con el axioma $par(n) = par(n \bmod p)$ el universo de modelos es vacío.

La solución 3 depende de la propiedad que adquieren los soldados en el tiempo: están vivos o muertos. La complejidad del algoritmo depende escencialmente de la operación de buscar un vecino vivo. Buscando reducir ésta complejidad encontramos el esquema directriz de la solución 4.

Solución 4

Vamos a basar ésta solución en el tipo abstracto anillo, cuya talla es decreciente en la medida en que los soldados sean eliminados. Se está incluyendo la operación de eliminación de los soldados.

Informalmente este algorítmo se basa en la eliminación de los soldados hasta que quede un solo soldado en el cículo. La determinación de esta condición de terminación puede hacerse de dos formas:

- Introduciendo la función talla del círculo y deteniendo la evaluación cuando esta talla sea unitaria.
- Si un elemento es igual a su sucesor, implica que el anillo es unitario. Debe tenerse en cuenta que ésta propiedad sólo es válida cuando en el anillo no hay elementos repetidos (es una realización de conjuntos).

IMPLEMENTACION

0 $J_2(A : anillo)$
1 si A es vaci'o $J_2 = 0$

```
2 sino
3   Candidato := PRIMERO(A)
4   hacer siempre
5       B := Candidato
6       Candidato := sucesor (Candidato)
7       si Candidato ≠ B entonces
8           eliminar Candidato,
9           declarando como
10          Candidato a su sucesor
11      sino exit
12  J2 := B
12  fin hacer
```

2.4 Propiedades derivables de la solución 4

La condición de terminación de esta implementación es la de obtener un anillo unitario. Vemos que esta solución permite, mediante transformaciones, modificarse para ser solución de $J_r(n)$, ya que basta cambiar la estrategia de eliminación (línea 6):

6' $candidato := sucesor^{r-1}(candidato)$

Esto hace fácil la evolución de la solución a una solución del problema más general.

Hemos mostrado una variedad de soluciones del problema de Josephus.

3 Conclusiones

Hemos presentado una función que se puede expresar informalmente en forma simple cuya implementaciónes varían en función del modelo utilizado para la presentación de los datos. Cada una de implementaciones es esencialmente diferente, dependiendo del modelo del cuál importa los operadores. Los modelos asociados corresponden a álgebras no isomorfas.

Las diferencias más resaltantes desde el punto de vista computacional son la velocidad de cómputo y la memoria requerida para la computación. Estas

implementaciones pueden presentarse como especificaciones, y cada una de ellas puede estar asociada a diferentes modelos.

Como elementos aislados, no se puede jerarquizar estas soluciones, dado que esa jerarquización depende de la arquitectura donde corra la aplicación y de los restantes operadores que sean usados en la aplicación.

Sin embargo la información obtenida durante las descripciones en modelos particulares puede contribuir a la reusabilidad del software desarrollado y para el test de desarrollos futuros. Una versión extendida de éste trabajo se encuentra en [ZOL].

References

[KNU2] R. Graham, D. Knuth, O.Patashnik Concrete Mathematics. A Foundation for Computer Science Addison Wesley Pub. Co. 1989.

[GAU] M. Bidoit, M.C. Gaudel PLUSS: Proposition pour un langage d'especification structurées Reporte interno Universidad de París Sud. 1985.

[ZOL] C.Zoltan Sobre especificaciones y modelos: un ejemplo Reporte N IC 1990-001 Departamento de Computación, Universidad Simón Bolívar Mayo 1990.