Clase de Rep y Abs: Soluciones

Algoritmos y Estructuras de Datos 2

1. Diccionario

La siguiente es la especificación del Tad Diccionario, restringida a observadores y generadores:

TAD DICCIONARIO(CLAVE, SIGNIFICADO)

observadores básicos

def?	: clave × dicc(clave, significado)	$\longrightarrow$	bool	
obtener	: clave c × dicc(clave, significado) d	$\longrightarrow$	significado	{def?(c, d)}

generadores

vacío	:	$\longrightarrow$	dicc(clave, significado)
definir	: clave × significado × dicc(clave, significado)	$\longrightarrow$	dicc(clave, significado)

axiomas $\forall d$: dicc(clave, significado), $\forall c, k$: clave, $\forall s$: significado

def?(c,vacío) $\equiv$ false

def?(c, definir(k, s, d) $\equiv$ $c = k \lor$ def?(c, d)

obtener(c, definir(k, s, d)) $\equiv$ **if** $c = k$ **then** s **else** obtener(c, d) **fi**

Fin TAD

Para representar el TAD DICCIONARIO se decidió utilizar la siguiente estructura:

diccionario **se representa con** estr, donde
estr es tupla ⟨ *claves*: lista(α),
significados: lista(β) ⟩

Donde cada par clave, significado se encuentra en el mismo índice de cada una de las listas.

a) Escribir en castellano el invariante de representación.

b) Escribir formalmente el invariante de representación.

c) Escribir formalmente la función de abstracción.

1.1. Invariante de representación en castellano

1. La longitud de ambas listas debe ser el mismo.
2. Claves no tiene elementos repetidos.

1.2. Invariante de representación

Rep : estr $\longrightarrow$ boolean

Rep(e) $\equiv$ true $\Longleftrightarrow$

longitud(e.claves) = longitud(e.significados) $\land$

$(\forall i, j : Nat)(i \neq j \land i \leq e.claves \land j \leq e.claves) \Rightarrow_L e.claves[i] \neq e.claves[j]$

1.3. Función de abstracción

Abs : estr e $\longrightarrow$ diccionario {Rep(e)}
($\forall e$: estr) Abs(e) $=_{\text{obs}}$ d: diccionario |

$(\forall c : \alpha)(c \in e.claves = def?(c, d)) \wedge_L$

$(\forall c_2 : \alpha)(c_2 \in e.claves \Rightarrow_L$

$(\exists i : Nat)(i \leq longitud(e.claves) \wedge_L e.claves[i] = c_2) \Rightarrow_L obtener(c_2, d) =_{\text{obs}} e.significados[i])$

2. Piratas y Ninjas

1erParcial 1erCuatrimestre 2015

La siguiente especificación modela un castillo donde conviven piratas y ninjas. Con frecuencia arriban al castillo nuevos piratas y ninjas, que nunca mueren ni se van. Por supuesto, cada tanto surgen peleas, que por tradición ancestral son siempre entre un pirata y un ninja. Los piratas y los ninjas se identifican con naturales unívocos: no hay dos piratas, ni dos ninjas, ni un pirata y un ninja que se identifiquen con el mismo número.

TAD CASTILLO

observadores básicos

piratas	: castillo	$\longrightarrow$	conj(nat)	
ninjas	: castillo	$\longrightarrow$	conj(nat)	
cantPeleas	: castillo c × nat p × nat n	$\longrightarrow$	nat	{$p \in$ piratas(c) $\wedge$ $n \in$ ninjas(c)}

generadores

crear	:	$\longrightarrow$	castillo	
llegaPirata	: castillo c × nat p	$\longrightarrow$	castillo	{$p \notin$ (piratas(c) $\cup$ ninjas(c))}
llegaNinja	: castillo c × nat n	$\longrightarrow$	castillo	{$n \notin$ (piratas(c) $\cup$ ninjas(c))}
pelean	: castillo c × nat p × nat n	$\longrightarrow$	castillo	{$p \in$ piratas(c) $\wedge$ $n \in$ ninjas(c)}

axiomas

piratas(crear)	$\equiv$	$\emptyset$	ninjas(crear)	$\equiv$	$\emptyset$
piratas(llegaPirata(c, p))	$\equiv$	Ag(p, piratas(c))	ninjas(llegaPirata(c, p))	$\equiv$	ninjas(c)
piratas(llegaNinja(c, n))	$\equiv$	piratas(c)	ninjas(llegaNinja(c, n))	$\equiv$	Ag(n, ninjas(c))
piratas(pelean(c, p, n))	$\equiv$	piratas(c)	ninjas(pelean(c, p, n))	$\equiv$	ninjas(c)

cantPeleas(llegaPirata(c, p'), p, n) $\equiv$ **if** $p = p'$ **then** 0 **else** cantPeleas(c, p, n) **fi**
cantPeleas(llegaNinja(c, n'), p, n) $\equiv$ **if** $n = n'$ **then** 0 **else** cantPeleas(c, p, n) **fi**
cantPeleas(pelean(c, p', n'), p, n) $\equiv$ **if** $p = p' \wedge n = n'$ **then** 1 **else** 0 **fi** + cantPeleas(c, p, n)

Fin TAD

Para representar el TAD CASTILLO se decidió utilizar la siguiente estructura:

castillo **se representa con** estr, donde
estr es tupla ⟨ *piratas*: conj(nat),
ninjas: conj(nat),
rivalesQueTuvo: dicc(nat, conj(nat)),
historialPeleas: secu(tupla ⟨p : nat, n : nat⟩) ⟩

donde *piratas* y *ninjas* representan los conjuntos de identificadores de piratas y ninjas, respectivamente, *rivalesQueTuvo* asocia a cada peleador (tanto piratas como ninjas, ya que todos los identificadores son distintos) con el conjunto de todos los rivales contra los que peleó al menos una vez, e *historialPeleas* tiene la secuencia de parejas ⟨pirata, ninja⟩ que se entreveraron en una pelea, en el orden en que éstas sucedieron.

a) Escribir en castellano el invariante de representación.
b) Escribir formalmente el invariante de representación.
c) Escribir formalmente la función de abstracción.

2.1. Invariante de representación

Rep(e) = 1 $\wedge$ 2 $\wedge_L$ 3 $\wedge_L$ 4 $\wedge$ 5 $\wedge_L$ 6 $\wedge$ 7

1) No hay Piratas que sean Ninjas (y viceversa):
$e.piratas \cap e.ninjas = \emptyset$

2) Todas las claves de e.RivalesQueTuvo son piratas o ninjas (y viceversa):
$claves(e.RivalesQueTuvo) = e.piratas \cup e.ninjas$

3) Los rivales de un pirata son ninjas y viceversa:
$(\forall p : Nat)(p \in e.piratas \Rightarrow_L obtener(p, e.RivalesQueTuvo) \subseteq e.ninjas) \wedge$
$(\forall n : Nat)(n \in e.ninjas \Rightarrow_L obtener(n, e.RivalesQueTuvo) \subseteq e.piratas)$

Nota: Sé que n y p están definidos en $e.RivalesQueTuvo$ por cláusula 2).

4) Reciprocidad de rivales en e.RivalesQueTuvo:
$(\forall i : Nat)(def?(i, e.RivalesQueTuvo) \Rightarrow_L$
$(\forall j : Nat)(j \in obtener(i, e.RivalesQueTuvo) \Rightarrow_L i \in obtener(j, e.RivalesQueTuvo)))$

Nota: Sé que j está definido en $e.RivalesQueTuvo$ por cláusula 2) y 3).

5) Tuplas válidas en e.HistorialPeleas: un pirata y un ninja :
$(\forall t :< Nat, Nat >)(esta?(t, e.HistorialPeleas) \Rightarrow t.p \in e.piratas \wedge t.n \in e.ninjas)$

6) Las peleas de e.HistorialPeleas figuran correctamente en e.RivalesQueTuvo:
$(\forall t :< Nat, Nat >)(esta?(t, e.HistorialPeleas) \Rightarrow t.p \in obtener(t.n, e.RivalesQueTuvo)$

Nota: Sé que $t.p$ está definido en $e.RivalesQueTuvo$ por 2) y 5) y sé que de 4) se deduce que $t.n \in obtener(t.p, e.RivalesQueTuvo)$.

7) Para cada luchador, los rivales que figuran en e.RivalesQueTuvo deben tener su pelea correspondiente en e.HistorialPeleas (básicamente la vuelta de 6):
$(\forall n : Nat)(n \in e.ninjas \Rightarrow_L$
$(\forall p : Nat)(p \in obtener(n, e.RivalesQueTuvo) \Rightarrow$
$(\exists t :< Nat, Nat >)(\pi_1(t) == p \wedge \pi_2(t) == n \wedge esta?(t, e.HistorialPeleas))))$

Nota: Sé que n está definido en $e.RivalesQueTuvo$ por cláusula 2).

2.2. Función de Abstracción

Abs(e): estr e $\rightarrow$ Castillo c $\{Rep(e)\}$
Abs(e) $\equiv c : Castillo \mid$
$piratas(c) =_{obs} e.piratas \wedge$
$ninjas(c) =_{obs} e.ninjas \wedge_L$
$(\forall n, p : Nat)(n \in e.ninjas \wedge p \in e.piratas \Rightarrow_L$
$cantPeleas(c, p, n) = contarPeleas(e.HistorialPeleas, p, n))$

contarPeleas: secu(<Nat, Nat>) x Nat p x Nat p $\rightarrow$ nat
contarPeleas(s,p,n) $\equiv$ if(vacia?(s)) then 0 else
(if($\pi_1(prim(s)) == p \wedge \pi_2(prim(s)) == n$) then 1 else 0) + contarPeleas(fin(s), p, n)