Invariante de Representación y Función de Abstracción

Clase práctica

Algoritmos y Estructuras de Datos II

16 de septiembre de 2017

Diferencias entre especificación y diseño

- En la etapa de especificación:
  - Nos ocupamos del '¿Qué?'
  - Lo explicamos usando TADs
  - Ese '¿Qué?' se explica bajo el paradigma funcional

Ejemplo: Conjunto en rango

TAD CONJRANG

Ig. obs.: $(\forall c_1, c_2 : \text{conjran})(c_1 =_{\text{obs}} c_2 \Leftrightarrow (\text{low}(c_1) = \text{low}(c_2) \land \text{up}(c_1) = \text{up}(c_2) \land (\forall n : \text{nat})((n \in c_1) = (n \in c_2)))$

observadores básicos

$\bullet \in \bullet \; : \; \text{nat} \times \text{conjran} \longrightarrow \text{bool}$

$\text{low} \; : \; \text{conjran} \longrightarrow \text{nat}$

$\text{up} \; : \; \text{conjran} \longrightarrow \text{nat}$

generadores

$\textit{emptyset} \; : \; \text{nat } \ell \times \text{nat } u \longrightarrow \text{conjran}$ $\{(\ell \leq u)\}$

$\text{Ag} \; : \; \text{nat } n \times \text{conjran } c \longrightarrow \text{conjran}$ $\{(\text{low}(c) \leq n \leq \text{up}(c))\}$

axiomas

$\forall c : \text{conjran}, \forall \ell, u, n, n' : \text{nat}$

$\text{low}(\emptyset(\ell, u)) \equiv \ell$

$\text{low}(\text{Ag}(n, c)) \equiv \text{low}(c)$

$\text{up}(\emptyset(\ell, u)) \equiv u$

$\text{up}(\text{Ag}(n, c)) \equiv \text{up}(c)$

$n \in \emptyset(\ell, u) \equiv \textit{false}$

$n \in \text{Ag}(n', c) \equiv (n = n') \lor (n \in c)$

Fin TAD

Diferencias entre especificación y diseño

- En la etapa de diseño:
  - Nos ocupamos del '¿Cómo?'
  - Lo explicamos con *módulos de abstracción*
  - Ese '¿Cómo?' se explica usando el paradigma imperativo
  - Tenemos un *contexto de uso* que nos fuerza a tomar decisiones respecto de la estructura.

Partes de un módulo

Un módulo de abstracción se divide en tres secciones:

- Interfaz: Es la sección accesible para los usuarios del módulo (Que bien pueden ser otros módulos). Aquí se detallan los servicios exportados. Cada operacion del módulo viene acompañado de:
  - Signatura.
  - Precondición.
  - Postcondición.
  - Complejidad Temporal.
  - Descripción (Importante!).
  - Aliasing (Lo vamos a ver más adelante).

Pertenece(in $\mathtt{n}$: $\mathtt{nat}$, in $\mathtt{c}$: $\mathtt{conjenrango}$) $\rightarrow$ $\mathtt{res}$: $\mathtt{bool}$

[Precondición]

[Postcondición]

Complejidad: $O(n)$

Descripción: Dado un elemento, devuelve verdadero si se encuentra en el conjunto. Caso contrario devuelve falso.

Partes de un módulo

- Representación: Esta sección no es accesible a los usuarios del módulo. Aquí se detalla la elección de estructura de representación y los algoritmos. Se justifica la elección de las estructuras así como la complejidad de los algoritmos.

La estructura elegida para representarlo es la siguiente:

Conjunto en rango se representa con *estr*
donde *estr* es
$<$ `low: nat, upper: nat, elems: secu(nat), min: nat` $>$

Partes de un módulo

- Servicios usados: Es la parte del módulo donde se detallan todas los supuestos sobre los servicios que exportan los otros módulos. Estos requisitos son los que justifican los análisis de complejidad que se hacen dentro de la sección Representación que a su vez justifican las promesas hechas en la interfaz.

En el ejemplo:

- Para chequear la pertenencia al conjunto utilizamos la operación *está?* de secuencia. El módulo secuencia nos permite, con complejidad lineal, saber si un elemento está en la misma o no.

Vinculación especificación y diseño

¿Cómo se relaciona todo esto con la especificación? Necesitamos saber dos cosas:

- ¿Qué valores pueden tomar las variables dentro de nuestra estructura para que esta sea válida?
  - Invariante de Representación (Rep): Es un predicado que expresa la sanidad de la estructura. Tiene que valer siempre en el momento inicial y final de cada función descrita en la interfaz.

$$\text{Rep:}\widehat{estr} \rightarrow \ boolean$$

- ¿Con qué instancia del TAD que estoy diseñando se vincula mi instancia de estructura de representación?
  - Función de Abstracción (Abs): Vincula la imagen abstracta de una estructura con el valor abstracto al que representa.

$$\text{Abs:}\widehat{\texttt{estr}} \rightarrow tipo\_abstracto\_representado$$

Recordar: La función $\widehat{\ }$ toma una estructura del mundo del diseño y nos devuelve automágicamente su correspondiente instancia abstracta del mundo de los TADs.

Invariante de Representación

Conjunto en rango se representa con *estr*
donde *estr* es
$<$ `low: nat, upper: nat, elems: secu(nat), min: nat` $>$

Cosas a tener en cuenta a la hora de escribir el Invariante de Representación:

- Restricciones de los generadores (¿Qué instancias puedo formar?).

En castellano:

- La cota inferior es menor o igual que la cota superior
- Todos los elementos del conjunto son mayores que la cota inferior y menores que la cota superior.

En lógica:

- e.low $\leq$ e.upper
- ($\forall$ n: Nat) está?(n, e.elems) $\Rightarrow$ e.low $\leq$ n $\leq$ e.upper

Invariante de Representación

Conjunto en rango se representa con *estr*
donde *estr* es
$<$ `low: nat, upper: nat, elems: secu(nat), min: nat` $>$

Cosas a tener en cuenta a la hora de escribir el Invariante de Representación:

- Decisiones de diseño.

En castellano:

La secuencia no tiene elementos repetidos.

En lógica:

$(\forall$ n: Nat$)$ está?(n, e.elems) $\Rightarrow_{\mathrm{L}}$ cantidadApariciones(n, e.elems) $= 1$

Invariante de Representación

Conjunto en rango se representa con *estr*
donde *estr* es
$<$ `low: nat, upper: nat, elems: secu(nat), min: nat` $>$

Cosas a tener en cuenta a la hora de escribir el Invariante de Representación:

- Coherencia en la información redundante.

En castellano:

- Si hay elementos en la secuencia, el mínimo es uno de ellos.
- El mínimo es efectivamente el mínimo del conjunto.

En lógica:

- $\neg\emptyset?(\text{e.elems}) \Rightarrow \text{e.min} \in \text{e.elems}$
- $(\forall\ \text{n: Nat})\ \text{está?}(\text{n, e.elems}) \Rightarrow \text{e.min} \leq \text{n}$

Función de Abstracción

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\texttt{conjuntoEnRango} \arrow[r, "\text{se representa con}"] \arrow[d, "\text{se explica con (\^{})}"'] & \texttt{estr} = \texttt{tupla}\langle ... \rangle \arrow[d, "\hat{}"'] \\
\textsc{conjRang} & \widehat{\texttt{estr}} = \mathsf{tupla}\langle ... \rangle \arrow[l, "\mathrm{Abs}"]
\end{tikzcd}
$$

Función de Abstracción

La función de Abstracción se puede escribir mapeando los observadores del TAD con las distintas partes de la estructura.

$$Abs : \widehat{\texttt{estr}}\ e \to \text{conjran}$$

$$Abs(e) = c \ / \ ((low(c) = e.lower) \wedge (up(c) = e.upper) \wedge$$
$$(\forall n : \text{nat})(n \in c \Leftrightarrow esta?(n, e.elems)))$$

Banda

Necesitamos modelar el comportamiento de un típico grupo de música para luego desarrollar un sistema que lo implemente. Los músicos y sus instrumentos se identifican por su nombre el cual es un STRING:

- Los músicos ingresan a la banda con un conjunto de instrumentos que poseen, y un cierto nivel de droga en sangre.
- El nivel de droga en sangre de los músicos aumenta a medida que se drogan.
- Un músico puede romper un instrumento, pero como hay códigos en la banda, sólo puede romper uno de los que trajo.
- Además nos interesa saber qué músicos están *lesionados*, es decir, los que rompieron todos los intrumentos que trajeron a la banda.
- A partir de la especificación y el diseño del problema, escribir el invariante de representación y la función de abstracción.

Ejercicio 1: Banda

TAD BANDA

observadores básicos

Músicos : Banda $\longrightarrow$ Conj(Músico)

#DrogaEnSangre : Banda b × Músico m $\longrightarrow$ nat $\{m \in \text{Músicos}(b)\}$

LoRompió? : Banda b × Músico m × Instrumento i $\longrightarrow$ Bool

$\{m \in \text{Músicos}(b) \wedge \text{i} \in \text{InstrumentosDeMúsico}(b,\ m)\}$

InstrumentosDeMúsico : Banda b × Músico m $\longrightarrow$ Conj(Instrumento)

$\{m \in \text{Músicos}(b)\}$

generadores

Iniciar : $\longrightarrow$ Banda

Drogarse : Banda b × Músico m $\longrightarrow$ Banda $\{\text{m} \in \text{Músicos(b)}\}$

AgregarMúsico : Banda × Músico × Conj(Instrumento) $\longrightarrow$ Banda $\{\text{m} \notin \text{Músicos(b)}\}$

RomperInstrumento : Banda b × Músico m × Instrumento i $\longrightarrow$ Banda

$\left\{\text{m} \in \text{Músicos(b)} \wedge_{\text{L}} \text{i} \in \text{InstrumentosDeMúsico(b, m)} \wedge \#\text{DrogaEnSangre(b, m)} \geq 9\right\}$

Fin TAD

Ejercicio 1: Banda

TAD BANDA

otras operaciones

Lesionados : Banda $\longrightarrow$ conj(Músico)

DameLesionados: Conj(Músico) *cm* $\times$ Banda *b* $\longrightarrow$ Conj(Músico) $\{\neg$vacia(cm)$\}$

rompioSusInstr : Músico *m* $\times$ Conj(Instrumento) *ci* $\times$ Banda *b* $\longrightarrow$ Bool

$\{$ci $\subseteq$ InstrumentosDeMúsico(m, b)$\}$

axiomas $\forall b$: Banda $\forall m, m'$: Músico $\forall i$: Instrumento

Músicos(AgregarMúsico(b, m)) $\equiv$ Ag(m, Músicos(b))

#DrogaEnSangre(m, Drogarse(b, m')) $\equiv$ **if** m = m' **then**
  1 + #DrogaEnSangre(b, m)
else
  #DrogaEnSangre(b, m)
fi

#DrogaEnSangre(m, AgMúsico(b, m')) $\equiv$ **if** m = m' **then**
  3
else
  #DrogaEnSangre(b, m)
fi

Fin TAD

Ejercicio 1: Banda

TAD BANDA

axiomas $\forall b$: Banda $\forall m, m'$: Músico $\forall i$: Instrumento

InstrumentosDeMúsico(m, AgMúsico(b,m',ci)) $\equiv$ **if** m = m' **then**
  ci
else
  InstMúsico(m, b)
fi

InstrumentosDeMúsico(m, RomperInstr(b,m',i)) $\equiv$ InstrumentosDeMúsico(m, b)

Fin TAD

Ejercicio 1: Banda

TAD BANDA

axiomas $\forall b$: Banda $\forall m, m'$: Músico $\forall i$: Instrumento

LoRompió?(m, i, AgregarMúsico(b, m', ci)) $\equiv$ **if** m = m' **then** False **else** LoRompió?(m, i, b) **fi**

LoRompió?(m, i, RomperInstr(b,m',i')) $\equiv$ (m = m' $\wedge$ i = i') $\vee$ LoRompió?(m, i, b)

Fin TAD

Ejercicio 1: Banda

TAD BANDA

axiomas $\forall b$: Banda $\forall m, m'$: Músico $\forall i$: Instrumento

```
Lesionados(b)              ≡ DameLesionados(Músicos(b), b)
DameLesionados(cm, b)      ≡ if rompioSusIntr(DameUno(cm), InstrumentosDeMúsi-
                             co(DameUno(cm), b), b) then
                                 Ag(DameUno(cm), DameLesionados(cm))
                             else
                                 DameLesionados(SinUno(cm))
                             fi
rompioSusInstr(m, ci, b)   ≡ if vacio(ci) then
                                 True
                             else
                                 if LoRompio?(DameUno(ci), m, b) then
                                     rompioSusIntr(m, SinUno(ci), b)
                                 else
                                     False
                                 fi
                             fi
```

Fin TAD

Ejercicio 1: Banda

Se decidió utilizar la siguiente estructura para representar el TAD.

Banda **se representa con** estr, donde

estr es tupla $\langle$*músicos*: conj(músico),
lesionados: conj(músico),
InstrumentosDeMúsico: dicc(músico, conj(instrumento))
InstrRotosDeMúsico: dicc(músico, conj(instrumento))
#DrogaEnSangre: dicc(músico, nat) $\rangle$

- *músicos* contiene a los músicos de la banda.
- *lesionados* contiene a los músicos que rompieron todos sus instrumentos.
- *InstrumentosDeMúsico* contiene para cada músico los instrumento que trajo a la banda.
- *InstrRotosDeMúsico* contiene para cada músico, cuáles de sus instrumentos rompió.
- *#DrogaEnSangre* contiene para cada músico su nivel de droga en sangre.

Ejercicio 2: Piratas y Ninjas

Ejercicio de Parcial (1er Cuatrimestre, 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 $\times$ nat p $\times$ nat n	$\longrightarrow$	nat

$\{p \in \text{piratas}(c) \land n \in \text{ninjas}(c)\}$

generadores

crear	:		$\longrightarrow$	castillo
llegaPirata	:	castillo c $\times$ nat p	$\longrightarrow$	castillo

$\{p \notin (\text{piratas}(c) \cup \text{ninjas}(c))\}$

llegaNinja	:	castillo c $\times$ nat n	$\longrightarrow$	castillo

$\{n \notin (\text{piratas}(c) \cup \text{ninjas}(c))\}$

pelean	:	castillo c $\times$ nat p $\times$ nat n	$\longrightarrow$	castillo

$\{p \in \text{piratas}(c) \land n \in \text{ninjas}(c)\}$

Fin TAD

TAD CASTILLO

axiomas

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

Ejercicio 3: Piratas y Ninjas

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

castillo **se representa con** estr, donde

estr es tupla $\langle$ *piratas*: conj(nat),
ninjas: conj(nat),
rivalesQueTuvo: dicc(nat, conj(nat)),
historialPeleas: secu(tupla $\langle p$: nat, n : nat$\rangle$) $\rangle$

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 $\langle$pirata, ninja$\rangle$ que se entreveraron en una pelea, en el orden en que éstas sucedieron.

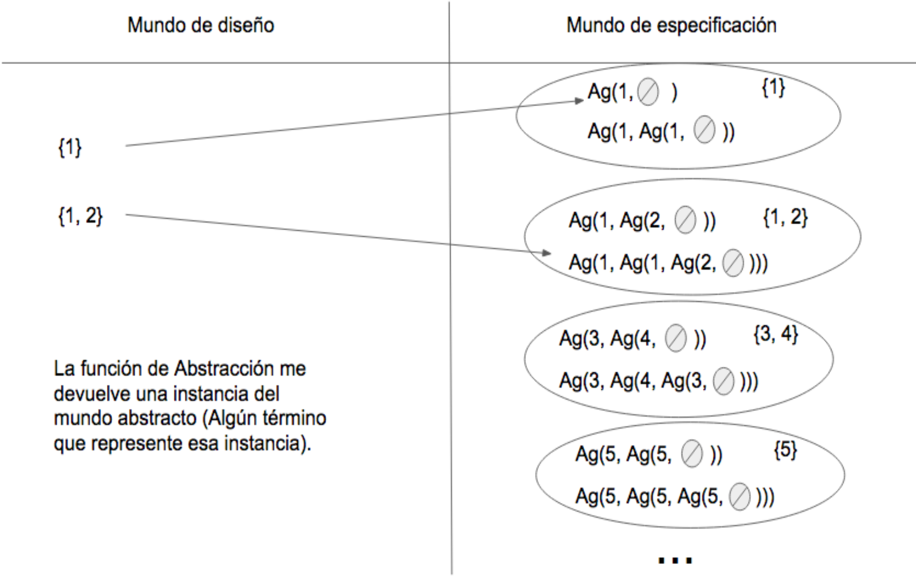
a) Escribir en castellano el invariante de representación.
b) Escribir formalmente el invariante de representación.
c) Escribir formalmente la función de abstracción.

Yapa para el final

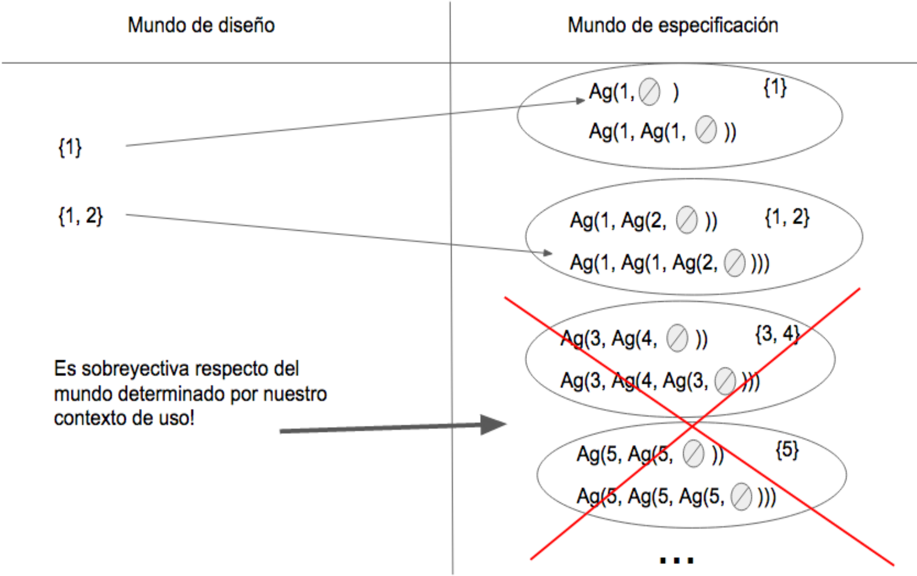
¿La función de Abstracción es sobreyectiva sobre el conjunto de términos?

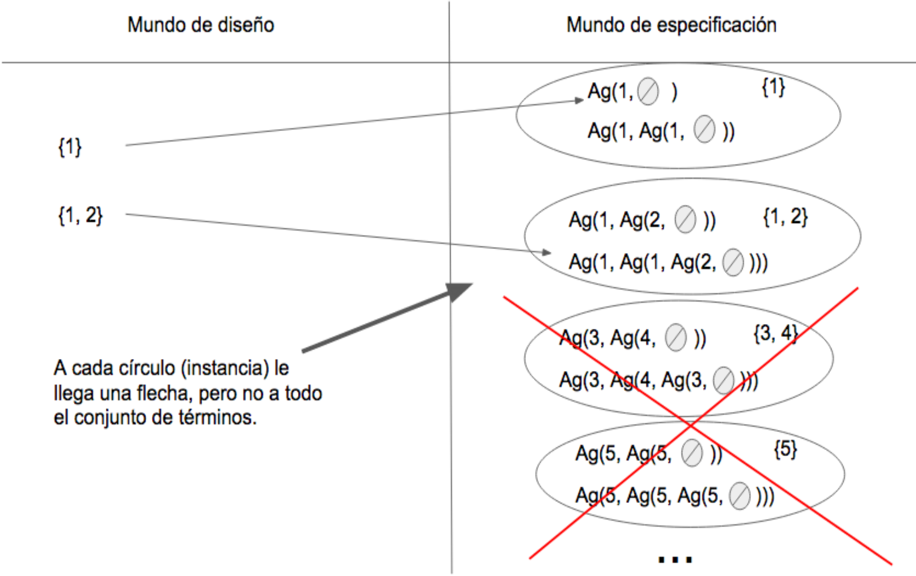
Mundo de diseño	Mundo de especificación
{1} {1, 2} Por nuestro contexto de uso, sólo representamos los conjuntos con un 1, o con un 1 y un 2.	

Mundo de diseño	Mundo de especificación
{1}	Ag(1, $\emptyset$) Ag(1, Ag(1, $\emptyset$)) {1}
{1, 2}	Ag(1, Ag(2, $\emptyset$)) Ag(1, Ag(1, Ag(2, $\emptyset$))) {1, 2}
	Ag(3, Ag(4, $\emptyset$)) Ag(3, Ag(4, Ag(3, $\emptyset$))) {3, 4}
En el mundo de especificación, tenemos potencialmente infinitas instancias del tipo representado.	Ag(5, Ag(5, $\emptyset$)) Ag(5, Ag(5, Ag(5, $\emptyset$))) {5}
	...



La función de Abstracción me devuelve una instancia del mundo abstracto (Algún término que represente esa instancia).


Mundo de diseño
Mundo de especificación
{1}
{1, 2}
Ag(1, ∅)
Ag(1, Ag(1, ∅))
{1}
Ag(1, Ag(2, ∅))
Ag(1, Ag(1, Ag(2, ∅)))
{1, 2}
Ag(3, Ag(4, ∅))
Ag(3, Ag(4, Ag(3, ∅)))
{3, 4}
Ag(5, Ag(5, ∅))
Ag(5, Ag(5, Ag(5, ∅)))
{5}
...
Es sobreyectiva respecto del mundo determinado por nuestro contexto de uso!


Mundo de diseño
Mundo de especificación
{1}
{1, 2}
Ag(1, ∅)
Ag(1, Ag(1, ∅))
{1}
Ag(1, Ag(2, ∅))
Ag(1, Ag(1, Ag(2, ∅)))
{1, 2}
Ag(3, Ag(4, ∅))
Ag(3, Ag(4, Ag(3, ∅)))
{3, 4}
Ag(5, Ag(5, ∅))
Ag(5, Ag(5, Ag(5, ∅)))
{5}
...
A cada círculo (instancia) le llega una flecha, pero no a todo el conjunto de términos.

:)

FIN