Un TAD está definido por un CONJUNTO DE VALORES Y OPERACIONES entre los mismos.

Las operaciones tienen una SEMÁNTICA descripta a través de AXIOMAS.

Cada TAD define una TEORÍA DE PRIMER ORDEN CON IGUALDAD.

Esto se logra especificando la SIGNATURA de las operaciones, con sus RESTRICCIONES sobre el dominio y los AXIOMAS que cumplen.

GENERADORES

Son las operaciones que permiten CONSTRUIR valores del tipo.

Deben poder CONSTRUIR CUALQUIER instancia.

Es ideal que sean MINIMALES.

OBSERVADORES BÁSICOS

Son las operaciones que permiten obtener información acerca de las instancias de un tipo.

Es ideal que sean MINIMALES.

AXIOMAS

Son las reglas que definen el COMPORTAMIENTO DE LAS FUNCIONES.

IGUALDAD OBSERVACIONAL

Define el criterio bajo el cual las instancias de un TAD son INDISTINGUIBLES.

Es un PREDICADO BINARIO entre las instancias de un tipo que indica CUANDO SON IGUALES.

Usualmente se define en término de los observadores básicos.

Se busca el conjunto minimal de operaciones que permina diferenciar entre todas las instancias de un tipo.

DISEÑO

Pasamos del paradigma FUNCIONAL al IMPERATIVO.

En esta etapa, además de respetar la SEMÁNTICA establecida, se deben considerar

ASPECTOS NO FUNCIONALES tales como:

- Eficiencia en TIEMPO Y ESPACIO
- ENCAPSULAMIENTO

En esta etapa se lleva a cabo:

- Proveer una REPRESENTACIÓN para los valores del tipo.
- Definir las FUNCIONES del tipo.

INTERFAZ

Define las operaciones exportadas junto con sus precondiciones y postcondiciones

lo cual establece la

CORRESPONDENCIA ENTRE OPERACIONES DE UN TAD Y SUS IMPLEMENTACIONES

Una función es REFERENCIALMENTE TRANSPARENTE si su resultado se puede determinar únicamente a partir de sus parámetros explícitos.

ESTRUCTURA DE REPRESENTACIÓN

Describe los valores sobre los cuales se representa el género implementado.

Debe poder representar todos los ELEMENTOS Y OPERACIONES de dicho género.

INVARIANTE DE REPRESENTACIÓN

Es un PREDICADO que determina qué instancias de la estructura representan

un elemento válido del género.

Esto sirve como documentación y es una CONDICIÓN NECESARIA para establecer

la RELACIÓN DE ABSTRACCIÓN.

También es una POSTCONDICIÓN IMPLÍCITA que se agrega a todas las operaciones,

garantizando que los algorirmos no rompas la estructura, así como una

PRECONDICIÓN, aseguramendo la validez de la misma.

Su validez como PRE Y POSTCONDICIÓN es lo que le da el carácter de INVARIANTE.

FUNCIÓN DE ABSTRACCIÓN

Es la herramienta que

permite vincular una estructura con ALGÚN valor abstracto al que representa.

- Su dominio: Instancias de la estructura que cumplen el invariante de rep.
- NO necesita ser INYECTIVA
- Debe ser SURYECTIVA sobre LAS CLASES DE EQUIVALENCIA de la igualdad observacional

La Estructura de Representación:

Describe los valores sobre los cuales se representará el género que se está implementando.

conjunto_semi_rápido **se representa con** estr

donde estr es tupla(*rápido*: arreglo [1 ... 100] de bool , *resto*: secu(nat) , *cardinal*: nat)

Invariante de Rep

$$Rep : estr \rightarrow bool$$

$$(\forall e : estr)\ Rep(e) \equiv true \Leftrightarrow \left[\begin{array}{l}\text{Condiciones que comparan} \\ \text{Obs del TAD con la estr}\end{array}\right]$$

$$Rep : \hat{estr} \rightarrow bool$$

$$(\forall e : \hat{estr}) Rep(e) \equiv true \Leftrightarrow$$

$$(1)(\forall n : nat)(\text{esta?}(n, e.resto) \ \Rightarrow \ n > 100) \wedge$$

$$(2)(\forall n : nat)(\text{cant\_apariciones}(n, e.resto) \leq 1) \wedge$$

$$(3) e.cant = long(e.resto) + \text{contar\_trues}(e.rapido)$$

Función de Abstracción

$$Abs : estr\ e \longrightarrow T \quad \{Rep(e)\}$$

ejemplo

$$\longrightarrow Conj\_semi\_rápido\ C$$

$$(\forall e : estr)\ Abs(e) =_{obs} C \ /$$

$$(\forall n : nat)\ n \in C \Leftrightarrow$$

$$\Big((n \leq 100 \land r.rápido[n]) \lor (n > 100 \land está?(n, e.resto)) \Big)$$

Abs: $\hat{estr}\ e \rightarrow$ conjunto_semi_rápido C {Rep(e)}

$$(\forall e : \hat{estr}) Abs(e) =_{\text{obs}} C \ /$$

$$(\forall n : nat)(n \in C \Leftrightarrow ((n \geq 100 \land r.rapido[n]) \lor$$

$$(n > 100 \land \text{está?}(n, e.resto)))$$

$f \in O(g)$:

$\forall n \in \mathbb{N}, \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}_{>0} /$

$$n \geq n_0 \Rightarrow f(n) \leq c \cdot g(n)$$

Teorema. *Si se tiene una relación de recurrencia de la siguiente forma:*

$$T(n) = \begin{cases} aT\left(\frac{n}{c}\right) + f(n) & si\ n > 1 \\ 1 & si\ n = 1 \end{cases}$$

Su fórmula cerrada es:

1. *Cuando* $f(n) \in \mathcal{O}(n^{\log_c a - \epsilon})$ *para algún* $\epsilon > 0$,

$$T(n) \in \Theta(n^{\log_c a})$$

2. *Cuando* $f(n) \in \Theta(n^{\log_c a})$,

$$T(n) \in \Theta(n^{\log_c a} \log n)$$

3. *Cuando* $f(n) \in \Omega(n^{\log_c a + \epsilon})$ *y* $af\left(\frac{n}{c}\right) \leq kf(n)$ *para* $k < 1$ *y* n *suficientemente grande,*

$$T(n) \in \Theta(f(n))$$

En los casos 1. y 3., se cumple $T(n) \in \Theta(f(n) + n^{\log_c a})$.

Agregar (in/out C: conj-semi_rápido , in e : nat)

Pre $\equiv \{ C =_{obs} C_0 \wedge e \notin C \}$

Post $= \{ C =_{obs} Ag(C_0, e) \}$

iAgregar (in/out C, in e : nat)

```
C.cant ++
if e < 100 then
    C.rápido [e] = true
else
    AgregarAtrás (C.resto, e)
```

Cota en altura de AVLs

Fibo. Trees : mínima cant. de nodos

Qvq :

$$Alt(T) \leq \log_k n$$

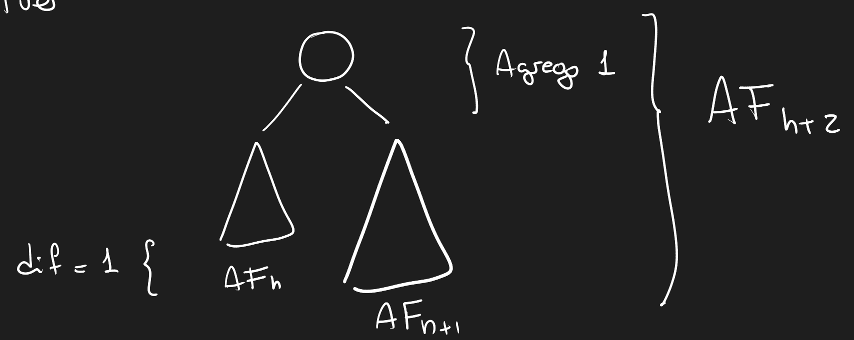
equiv

$$k^{Alt(T)} \leq n$$

Si AF_h es Arbol de Fibo de órden (altura) h

$$\Rightarrow |AF_{h+2}| = |AF_h| + |AF_{h+1}| + 1$$

Pues



Como la secu de Fibona. crece exponencialmente

$\Rightarrow |AF_h| \in \Omega(k^h)$ para algún k

y como por def $|T| \geq |AF_h|$ $\forall$ arbol AVL T de altura h

$$\Rightarrow Alt(T) \in O(\log n)$$

Build Heap (Algo de Floyd)

Cota:

Arbol binario de n nodos $\left(n = 2^{h+1} - 1\right)$

- $\frac{n+1}{2}$ hojas
- $\frac{n+1}{4}$ nodos en penultimo nivel (1 nivel por encima de las hojas)
- $\frac{n+1}{8}$ nodos en ante-penultimo nivel (2 niveles por encima de las hojas)

$\vdots$

Altura

$$\underbrace{\left(1 \cdot \frac{n+1}{2} + 2 \cdot \frac{n+1}{2.2} + 3 \cdot \frac{n+1}{2.4} + \cdots + h \cdot \frac{n+1}{2.2^h}\right)}_{\text{Por cada nivel del arbol}}$$

$$\sum_{h=0}^{\log n} \frac{n.h}{2^{h+1}} = n. \underbrace{\sum_{h=0}^{\log n} \frac{h}{2^{h+1}}}_{\text{Serie acotada por una constante}}$$

$$\leq c.\, n \qquad c \in \mathbb{R}_{>0}$$

$\therefore$ Algo $\in O(n)$