Teo. 2

otras formas de especificar

- Métodos informales
  - ↳ Menor costo inicial
  - ↳ No permite encontrar errores
  - ↳ costo a mediano plazo
- Métodos formales
  - Algebraicos (ej TADs) ←
  - Operacionales
  - Basados en estados

odularidad

TAD Conjunto

TAD Conjunto(α)

parámetros formales
  géneros α
  → **operaciones** No se requiere ninguna en particular.

géneros conj(α)

observadores básicos
  $\bullet \in \bullet$: $\alpha \times$ conj(α) $\longrightarrow$ bool

generadores
  $\emptyset$: $\longrightarrow$ conj(α)
  Ag : $\alpha \times$ conj(α) $\longrightarrow$ conj(α)

operaciones
  $\emptyset?$: conj(α) $\longrightarrow$ bool
  $\#$: conj(α) $\longrightarrow$ nat
  $\bullet - \{\bullet\}$: conj(α) $\times \alpha \longrightarrow$ conj(α)
  → dameUno : conj(α) $c \longrightarrow \alpha$ $\{\neg\emptyset?(c)\}$
  → sinUno : conj(α) $c \longrightarrow$ conj(α) $\{\neg\emptyset?(c)\}$

Fin TAD

α puede ser "cualquier cosa"

aridad = 2

- Vista la parte Sintáctica de TAD,

Pasamos a la parte re-piola ✓,

Semántica del TAD

axiomas

$(\forall\ c, d : \text{conj}(\alpha))\ ,\ (\forall\ a, b : \alpha)$

$a \in \emptyset$	$\equiv$	$false$
$a \in Ag(b,\ c)$	$\equiv$	$(a \equiv b) \vee (a \in c)$
$\emptyset?(\emptyset)$	$\equiv$	$true$
$\emptyset?(Ag(b,\ c))$	$\equiv$	$false$
$\#(\emptyset)$	$\equiv$	0
$\#(Ag(a,\ c))$	$\equiv$	$1 + \#(c - \{\ a\ \})$
$\emptyset - \{\ a\ \}$	$\equiv$	$\emptyset$
$Ag(a,\ c) - \{\ b\ \}$	$\equiv$	**if** $a \equiv b$ **then** $c - \{\ b\ \}$ **else** $Ag(a,\ c - \{\ b\ \})$ **fi**
$dameUno(c) \in c$	$\leftarrow$	$c \neq \emptyset$
$sinUno(c)$	$\equiv$	$c - \{\ dameUno(c)\ \}$

ese ≡ corresponde a la Semántica del tipo abstracto α (pues a y b son del tipo α)

tl;dr: "equivalente" es relativo a lo que comparemos.

Para ∨ también.

notar que dameUno y sinUno devuelven o sacan el mismo c

(Emme)

"[...] hay cosas que parecen tácitas, pero si no las ponemos por escrito podemos caer en problemas a la hora de axiomatizar"

! Las operaciones son funciones

↳ evitar "sobre especificar"

si el 1° valor de la sec. S es 3 ↳ hace "una cosa" cualquier otro valor se guarda en a

! No hay pattern matching

$f(3 \cdot 5) =$ una cosa

$f(a \cdot 5) =$ otra cosa ← a puede valer 3

¡No queremos subespecificar

¿Qué tanto especificamos?

me voy para arriba...

- Regla práctica:
  - ↳ Axiomatizar los observadores básicos sobre todos los generadores no restringidos.

Buen intento intentando explicar ESTO!

- platos_de_cierto_precio(r, c, x) ≡ **if** ∅?(c) **then** ∅ **else if** precio(DameUno(c), r)=x **then** Ag(DameUno(c), platos_de_cierto_precio(r, SinUno(c), x)) **else** platos_de_cierto_precio(r, SinUno(c), x) **fi fi**

- platos_de_x_precio (r, c, x) ≡ (r: restaurante (nombre / ID); x: precio del plato a buscar; c: conjunto de platos)

  if ∅?(c) then

    ∅ ← devuelve el conj. vacío

  else

    if precio(DameUno(c), r) = x then (DameUno(c): Dame 1 plato del conj; precio(...) = x: ¿tiene precio x?)

      Ag(DameUno(c), platos_de_x_precio(r, SinUno(c), x)) (Ag: lo agrego a ...; platos_de_x_precio(r, SinUno(c), x): llamado recursivo sin ese plato; SinUno(c): lo saco)

    else

      platos_de_x_precio(r, SinUno(c), x) (mismo llamado recursivo, pero no agregué nada al conjunto que devuelve.)

    fi

  fi

Escribiremos

"$=$" en lugar de "$\equiv$"

Interpretamos $a = b$ ("reescribo")

"a semánticamente equivalente a b"

y no

"a sintácticamente igual a b"

Conviene

conjunto de { generadores / observadores } mínimal.

Polisemia

↳ 1 palabra - varios significados

" en el mismo texto, pero más adelante "

Igualdad observacional

- "mismo comportamiento"
- igualdad entre instancias de un tipo abs. de dato

$$Ag(1, Ag(2, \phi)) \underset{obs}{=} Ag(2, Ag(1, \phi))$$

Congruencia

f es congruente wrt una relación de equivalencia $\sim$ sii

$$(\forall x, y)(x \sim y \iff f(x) \sim f(y))$$