

1. Módulo ÁrbolCategorías

Interfaz

se explica con: `ÁRBOLCATEGORÍAS, ITERADOR UNIDIRECCIONAL(CATEGORÍA)`.

géneros: `acat, datoscat, intercat`.

Operaciones básicas de árbol de categorías

CREARÁRBOL(*in raiz: categoria*) $\rightarrow res : acat$
Pre $\equiv \{\neg vacía?(raiz)\}$
Post $\equiv \{res =_{obs} nuevo(raiz)\}$
Complejidad: $\Theta(|raiz|)$
Descripción: crea un árbol nuevo cuya categoría raíz es *raiz*.

NOMBRECATEGORÍARAÍZ(*in ac: acat*) $\rightarrow res : categoria$
Pre $\equiv \{true\}$
Post $\equiv \{res =_{obs} raíz(ac)\}$
Complejidad: $\Theta(1)$
Descripción: devuelve el nombre de la categoría raíz de *ac*.

AGREGARCATEGORÍA(*in hija: categoria, in padre: categoria, in/out ac: acat*)
Pre $\equiv \{ac =_{obs} ac_0 \wedge está?(padre, ac) \wedge \neg vacía?(hija) \wedge \neg está?(hija, ac)\}$
Post $\equiv \{ac =_{obs} agregar(ac_0, padre, hija)\}$
Complejidad: $\Theta(|padre| + |hija|)$
Descripción: agrega la categoría *hija* como hija de la categoría *padre*.

CREARITERCAT(*in padre: categoria, in ac: acat*) $\rightarrow res : intercat$
Pre $\equiv \{está?(padre, ac)\}$
Post $\equiv \{alias(esPermutacion?(SecuSuby(res), hijos(ac, padre))) \wedge vacia?(Anteriores(res))\}$
Complejidad: $\Theta(|padre|)$
Descripción: devuelve un iterador unidireccional de las categorías hijas directas de la categoría *padre*.

DUDA: ¿Puedo tratar a *res* acá directamente como un `itConj(α)` en la expresión `SecuSuby(res)?`

DUDA: ¿Hay que extender el TAD ÁrbolCategorías como en el apunte de módulos básicos para poder especificar la operación `esPermutacion??`

IDCATEGORÍAPORNOMBRE(*in c: categoria, in ac: acat*) $\rightarrow res : nat$
Pre $\equiv \{está?(c, ac)\}$
Post $\equiv \{res =_{obs} id(ac, c)\}$
Complejidad: $\Theta(|c|)$
Descripción: devuelve el *id* de la categoría *c*.

Operaciones de datos de categoría

DUDA: ¿Hace falta crear un TAD para el género `datoscat` para poder especificar las pre y postcondiciones de las funciones a continuación?

OBTENERID(*in dc: datoscat*) $\rightarrow res : nat$
Pre $\equiv \{???\}$
Post $\equiv \{???\}$
Complejidad: $\Theta(1)$
Descripción: devuelve el *id* de la categoría asociada a *dc*.

OBTENERNOMBRE(*in dc: datoscat*) $\rightarrow res : nat$
Pre $\equiv \{???\}$
Post $\equiv \{???\}$
Complejidad: $\Theta(1)$
Descripción: devuelve el nombre de la categoría asociada a *dc*.

OBTENERPADRE(*in dc: datoscat*) $\rightarrow res : puntero(datoscat)$
Pre $\equiv \{???\}$
Post $\equiv \{???\}$
Complejidad: $\Theta(1)$

Descripción: devuelve un puntero a los datos de la categoría padre asociada a dc .

OBTENERHIJOS(**in** dc : **datoscat**) $\rightarrow res$: **conj**(**puntero**(**datoscat**))

Pre $\equiv \{???\}$

Post $\equiv \{???\}$

Complejidad: $\Theta(1)$

Descripción: devuelve un conjunto de punteros a los datos de las categorías hijas directas asociadas a dc .

Operaciones de iterador de categorías

HAYMÁS?(**in** it : **itercat**) $\rightarrow res$: **bool**

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{HayMás?}(it)\}$

Complejidad: $\Theta(1)$

Descripción: devuelve **true** si y sólo si en el iterador todavía quedan elementos para avanzar.

ACTUAL(**in** it : **itercat**) $\rightarrow res$: **puntero**(**datoscat**)

Pre $\equiv \{\text{HayMás?}(it)\}$

Post $\equiv \{res =_{\text{obs}} \text{Actual}(it)\}$

Complejidad: $\Theta(1)$

Descripción: devuelve el elemento actual del iterador.

AVANZAR(**in/out** it : **itercat**)

Pre $\equiv \{it =_{\text{obs}} it_0 \wedge \text{HayMás?}(it)\}$

Post $\equiv \{res =_{\text{obs}} \text{Avanzar}(it_0)\}$

Complejidad: $\Theta(1)$

Descripción: avanza el iterador a la posición siguiente.

Representación

Representación de árbol de categorías

acat se representa con estr_acat

donde **estr_acat** es **tupla**(**raíz**: **puntero**(**datoscat**), **categorías**: **dicctrie**(**datoscat**))

Invariante de representación:

1. raíz no puede ser nulo.
2. raíz tiene que estar en el diccionario de categorías.
3. raíz tiene que tener id 1.
4. para todas las categorías en el diccionario:
 - a) la categoría no puede ser nula.
 - b) el nombre de la categoría deber ser igual a su clave en el diccionario.
 - c) el id de la categoría debe estar en rango.
 - d) dos categorías no pueden tener el mismo id.
 - e) los hijos de la categoría tienen que estar en el diccionario de categorías.
 - f) el padre es nulo si y sólo si la categoría es la raíz.
 - g) si el padre no es nulo, tiene que estar en el diccionario de categorías.
 - h) si el padre no es nulo, la categoría está entre los hijos del padre.
 - i) si el padre no es nulo, el id de la categoría debe ser superior al del padre.

Rep : **estr_acat** \rightarrow **bool**

$\text{Rep}(e) \equiv \text{true} \iff$
 (1) $\neg(e.\text{raíz} =_{\text{obs}} \text{NULL}) \wedge_L$
 (2) $\text{def?}(e.\text{raíz} \rightarrow \text{nombre}, e.\text{categorías}) \wedge_L \text{obtener}(e.\text{raíz} \rightarrow \text{nombre}, e.\text{categorías}) =_{\text{obs}} e.\text{raíz} \wedge$
 (3) $e.\text{raíz} \rightarrow \text{id} =_{\text{obs}} 1 \wedge$
 (4) $(\forall c: \text{categoria})(\text{def?}(c, e.\text{categorías}) \Rightarrow_L ($
 (4a) $\neg(\text{obtener}(c, e.\text{categorías}) =_{\text{obs}} \text{NULL}) \wedge_L$
 (4b) $\text{obtener}(c, e.\text{categorías}) \rightarrow \text{nombre} =_{\text{obs}} c \wedge_L$
 (4c) $1 \leq \text{obtener}(c, e.\text{categorías}) \rightarrow \text{id} \wedge \text{obtener}(c, e.\text{categorías}) \rightarrow \text{id} \leq \#(\text{claves}(e.\text{categorías})) \wedge$
 (4d) $(\forall c': \text{categoria})(\text{obtener}(c, e.\text{categorías}) \rightarrow \text{id} =_{\text{obs}} \text{obtener}(c', e.\text{categorías}) \rightarrow \text{id} \iff c =_{\text{obs}} c') \wedge$
 (4e) $(\forall h: \text{puntero}(\text{datoscat}))(h \in \text{obtener}(c, e.\text{categorías}) \rightarrow \text{hijos} \Rightarrow_L$
 $\text{def?}(h \rightarrow \text{nombre}, e.\text{categorías})) \wedge_L$
 (4f) $\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} =_{\text{obs}} \text{NULL} \iff c =_{\text{obs}} e.\text{raíz} \rightarrow \text{nombre} \wedge_L$
 (4g) $\neg(\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} =_{\text{obs}} \text{NULL}) \Rightarrow_L$
 $\text{def?}(\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} \rightarrow \text{nombre}, e.\text{categorías}) \wedge_L$
 (4h) $\neg(\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} =_{\text{obs}} \text{NULL}) \Rightarrow_L$
 $\text{obtener}(c, e.\text{categorías}) \in \text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} \rightarrow \text{hijos}) \wedge$
 (4i) $\neg(\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} =_{\text{obs}} \text{NULL}) \Rightarrow_L$
 $\text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} \rightarrow \text{id} < \text{obtener}(c, e.\text{categorías}) \rightarrow \text{id}))$

DUDA: ¿Está bien acceder a los campos de `datoscat` como si fuera una tupla? Notar que se usa el género `datoscat` en vez de su estructura de representación `estr_datoscat`.

DUDA: Para acortar el `Rep`, ¿puedo declarar variables dentro del mismo? Ejemplo:

$$\begin{aligned}
 x =_{\text{obs}} \text{obtener}(c, e.\text{categorías}) \wedge \neg(x \rightarrow \text{padre} =_{\text{obs}} \text{NULL}) \Rightarrow_L \\
 (\text{def?}(x \rightarrow \text{padre} \rightarrow \text{nombre}, e.\text{categorías}) \wedge_L x \in x \rightarrow \text{padre} \rightarrow \text{hijos})
 \end{aligned}$$

$\text{Abs} : \text{estr_acat } e \longrightarrow \text{acat} \quad \{\text{Rep}(e)\}$
 $\text{Abs}(e) =_{\text{obs}} \text{ac} : \text{acat} \mid \text{categorias}(\text{ac}) =_{\text{obs}} \text{claves}(e.\text{categorías}) \wedge_L$
 $\text{raíz}(\text{ac}) =_{\text{obs}} e.\text{raíz} \rightarrow \text{nombre} \wedge$
 $(\forall c: \text{categoria})(c \in \text{claves}(e.\text{categorías}) \Rightarrow_L ($
 $\text{padre}(\text{ac}, c) =_{\text{obs}} \text{obtener}(c, e.\text{categorías}) \rightarrow \text{padre} \rightarrow \text{nombre} \wedge$
 $\text{id}(\text{ac}, c) =_{\text{obs}} \text{obtener}(c, e.\text{categorías}) \rightarrow \text{id}))$

Representación de datos de categoría

`datoscat` se representa con `estr_datoscat`

donde `estr_datoscat` es tupla(*id*: `nat` , *nombre*: `categoria` , *padre*: `puntero(datoscat)` , *hijos*:
`conj(puntero(datoscat))`)

DUDA: ¿Hace falta crear un TAD nuevo para poder expresar el `Rep` y `Abs` del género `datoscat`? De ser así, ¿tengo que agregar dicho TAD en la lista 'se explica con'?

Representación de iterador de categorías

`itercat` se representa con `itConj(puntero(datoscat))`

DUDA: ¿Está bien la estructura de representación elegida?

DUDA: ¿Qué usamos para expresar el `Rep` y el `Abs` en este caso? Dado que se trata de un `itConj(α)`, suena razonable pensar en usar el `Rep` y `Abs` de este iterador definidos en el módulo `Conjunto Lineal(α)`.

Algoritmos

Pendiente.

2. Módulo LinkLinkIt

Interfaz

se explica con: `LINKLINKIT, ITERADOR UNIDIRECCIONAL(LINK)`.

géneros: lli, datoslink, iterlli.

Operaciones básicas del sistema

Pendiente.

Operaciones del iterador

Pendiente.

Representación

Representación del sistema

Pendiente.

Representación del iterador

Pendiente.

Algoritmos

Pendiente.

3. Módulo Diccionario Trie(α)

Interfaz

parámetros formales

géneros α

función COPIAR($\text{in } a : \alpha \rightarrow res : \alpha$)

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} a\}$

Complejidad: $\Theta(\text{copy}(a))$

Descripción: función de copia de α 's

se explica con: DICCIONARIO(STRING, α), ITERADOR UNIDIRECCIONAL(α).

géneros: dicctrie(α), iterdicctrie(α).

Operaciones básicas de diccionario trie

Pendiente.

Operaciones del iterador

Pendiente.

Representación

Representación de diccionario trie

Pendiente.

Representación del iterador

Pendiente.

Algoritmos

Pendiente.