Tema 3
Concepto y Especificación de Tipos Abstractos de Datos

Pablo Sánchez

Dpto. Matemáticas, Estadística y Computación
Universidad de Cantabria
Santander (Cantabria, España)
p.sanchez@unican.es

Objetivos

Objetivos

1. Conocer el concepto de tipo abstracto de datos.
2. Entender la diferencia y relación entre especificación e implementación.
3. Entender la utilidad de las especificaciones algebraicas.
4. Ser capaz de interpretar y entender una especificación algebraica.

Bibliografía Básica


Ricardo Peña (2005).
Diseño de Programas: Formalismo y Abstracción.
Pearson Educacion, 3 edition.


Xavier Franch (1999).
Estructuras de Datos: Especificación, Diseño e Implementación.
Ediciones Universidad Politécnica de Cataluña, 3 edition.

Concepto de Abstracción

Abstracción

Proceso por el cual se resaltan o destacan los detalles relevantes para un cierto propósito mientras se ignoran u ocultan los irrelevantes para tal propósito.

Conjunto de Fechas

3/Marzo/1989

Posibles Representaciones

```
Fecha ES TUPLA
    dia: RANGO [1..31] DE NATURAL;
    mes: RANGO [1..12] DE NATURAL;
    año: NATURAL;
FINTUPLA

// Cuento los dias desde el nacimiento de cristo
Fecha ES NATURAL;

// Cuento los dias desde el comienzo del año
Fecha ES TUPLA
    ndias: RANGO [0..365] DE NATURAL;
    año  : NATURAL;
FINTUPLA
```

Especificación vs Implementación

Especificación Sw

Determinar qué debe hacer un elemento software, sin detallar cómo lo hace.

Implementación Sw

Determinar cómo se satisface una determinada especificación.

Para una especificación puede haber varias (incluso infinitas) implementaciones que la satisfacen.

Especificación Formal de Algoritmos

Especificación con Pre y Postcondiciones

Una especificación de un algoritmo A es una terna $\{Q\}S\{R\}$ donde Q y R son predicados y S es la cabecera (o signatura) para el algoritmo A.

1. El predicado $\{Q\}$ se denomina *precondición* y sus únicas variables libres son los parámetros de entrada de S.
2. El predicado $\{R\}$ se denomina *postcondición* y sus únicas variables libres son los parámetros de S.
3. El significado de dicha terna es que si el algoritmo A comienza con una asignación de variables que satisface Q, A termina y lo hace en un estado que satisface R.

Concepto de Tipo Abstracto de Datos

Tipo Abstracto de Datos (TAD)

Un *tipo abstracto de datos* es un tipo, es decir un conjunto de individuos más una serie de operaciones (básicas) aplicables a dicho conjunto de individuos, que se define de forma independiente a su representación.

Una especificación de un TAD debe indicar:

1. Qué individuos representa (o pertenecen a) dicho tipo;
2. Qué operaciones que se pueden aplicar sobre tales individuos;
3. La *semántica* y propiedades de tales operaciones.

Especificación Algebraica de TADs

Especificación algebraica

Una *especificación algebraica* es un par $SPEC = (SIG, E)$, donde SIG es una signatura y E un conjunto de ecuaciones con respecto a SIG.

Signatura

Una signatura es un par $SIG = (S, OP)$, donde S es un conjunto de dominios y OP es un conjunto de funciones sobre los dominios de S

Las operaciones y ecuaciones de una $SPEC$ determinan:

1. Qué valores del TAD podemos construir;
2. Qué operaciones se pueden aplicar a un valor del TAD;
3. Qué hace cada operación y qué propiedades tiene.

Valores Generados por una Esp. Algebraica

Términos de un TAD

Dada una signatura $SIG = (S, OP)$ y un conjunto X de variables con respecto a SIG, el soporte del *SIG-álgebra de términos abiertos*, denotada $T_{SIG}(X)$ se define como, para todo $s \in S$:

1. $\forall (op :\rightarrow s) \in OP, op \in T_{SIG}$
2. $\forall (op : s_1 \dots s_n) \in OP, \forall t_1 : s_1 \in T_{SIG} \dots \forall t_n : s_n \in T_{SIG}$,
   $t_1 \dots t_n \models Pre(op)$;
   $op(t_1, \dots, t_n) \in T_{SIG}$,
   donde $Pre(op)$ es el conjunto de las precondiciones de op.

Pueden existir más valores de los deseados sino los agrupamos de algún modo.

Álgebra Asociada a un TAD

Álgebra definida por SPEC

Dada una especificación $SPEC = (SIG, E)$, el álgebra definida por $SPEC$, denotada T_{SPEC}, es la siguiente SIG-Álgebra:

1. Soporte: $T_{SPEC} = T_{SIG}/\equiv_E$, donde $\equiv_E$ es la relación de equivalencia inducida por las ecuaciones. Dado $t \in T_{SIG}$, su clase de equivalencia se denota $[t]$.
2. Operaciones:
   - $\forall (op :\rightarrow s) \in OP, op^{T_{SPEC}} = [t]$
   - $\forall (op : s_1 \ldots s_n) \in OP$, $[t_1], \ldots, [t_n] \models Pre(op)$;
     $op^{T_{SPEC}}([t_1], \ldots, [t_n]) = [op^{T_{SPEC}}(t_1, \ldots, t_n)]$

Término Canónico

El término canónico de una clase de equivalencia es el término más *simple* de una clase de equivalencia.

Modelo de una Especificación Algebraica

Una especificación algebraica $SPEC = (SIG, E)$ tiene como modelo un álgebra A tal que informalmente:

1. Hay una función total $f : T_{SPEC} \rightarrow A$.
2. Posee un equivalente para cada operación $op \in SIG$.
3. Las ecuaciones de $SPEC$ se satisfacen en A.
4. No posee basura: $\forall t \in A$, t debe poder ser generado por las operaciones de SIG.
5. No posee confusión: $\forall t_1, t_2 \in T_{SPEC}, t_1 \neq t_2; f(t_1) \neq f(t_2)$

Clases de Operaciones de una Especificación Alg.

Operaciones Constructoras Su resultado es del tipo de interés.

Operaciones Observadoras Tienen como parámetro al tipo de interés, pero el resultado no es del tipo de interés.

Operaciones Generadoras Mínimo subconjunto de las constructoras con las que es posible generar cualquier término del tipo.

- Se dicen *libres* si no hay confusión entre los términos generados.
- Se dicen *no libres* en caso contrario.

Elaboración de Especificaciones Algebraicas

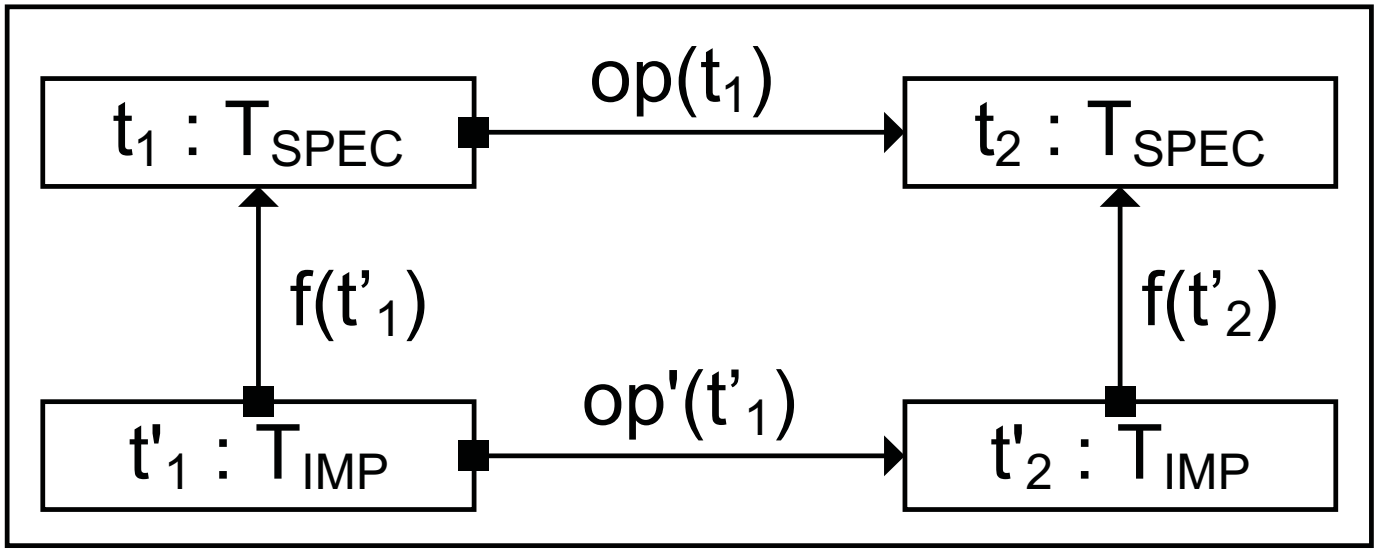
1. Seleccionar el conjunto de generadoras $Gen(s)$.
2. Si $Gen(s)$ no fuesen libres, escribir las ecuaciones necesarias para evitar la confusión.
3. Identificar el patrón de los términos canónicos.
4. Escribir ecuaciones para el resto de operaciones, al menos una por operación y patrón de los términos canónicos.
5. Estas ecuaciones deben garantizar que las constructoras son reducibles a términos canónicos.

Ventajas de las Especificaciones Algebraicas

1. Permiten especificar con precisión la sintaxis y comportamiento de un TAD.
2. Son ejecutables, por lo que sirven para la creación de prototipos.
3. Permiten verificar formalmente propiedades de los TADs.,

Relación Especificación-Implementación

- Los valores del tipo se *simulan* mediante las construcciones y tipos de un lenguaje de programación.
- Existe función $f : T_{IMP} \rightarrow T_{SPEC}$, denominada *función de abstracción*, que a cada valor concreta asigna su valor abstracto.
- f es sobreyectiva, no necesariamente inyectiva, parcial y homeomorfa con respecto a la especificación.



Corrección de una Implementación

1. Definir el *invariante de la representación* I_{IMP} (define los valores válidos).
2. Definir la relación de equivalencia para la representación $\equiv_{IMP} \subseteq T_{IMP} \times T_{IMP}$
3. El álgebra $(T_{IMP} \models I_{IMP})/\equiv_{IMP}$ debe ser isomorfa a $T_{SPEC}/\equiv_E$
4. Consecuencias:
   - El invariante es precondición de cada operación implementada.
   - El invariante es postcondición de cada operación implementada.

¿Qué tengo que saber de todo esto?

1. Conocer la diferencia entre especificación y diseño/implementación.
2. Conocer el concepto de Tipo Abstracto de Datos (TAD).
3. Ser consciente de la importancia de especificar con precisión.
4. Saber qué es una especificación algebraica y sus ventajas.
5. Saber leer y entender una especificación algebraica.
6. Saber qué es el *invariante* y la *relación de equivalencia* de la representación.

Referencias I


Ricardo Peña (2005).
Diseño de Programas: Formalismo y Abstracción.
Pearson Educacion, 3 edition.


Xavier Franch (1999).
Estructuras de Datos: Especificación, Diseño e Implementación.
Ediciones Universidad Politécnica de Cataluña, 3 edition.