API: Specifica delle Operazioni

Sommario: 17 Maggio, 2016

- API in OCaml
- API in Java
- Forma del Valori Astratti e Specifica delle operazioni
- Applicazioni: Design by Contract

API in OCaml

- **API** deve contenere tutto quello che chi usa i valori astratti
  - introdotti dall'API e
  - Implementati da un ADT

  deve conoscere per usarli in modo appropriato
- **OCaml** ha specifici moduli per esprimere una API

  module type *ModuleName* =
  sig type *TypeName*
  val *op1* ... **val** *opn*

  end;;
- Ma sulla specifica delle operazioni le API OCaml si limitano alla sola signature

```
module type RELAZIONE =
 sig type ('a,'b) relazione
  val relazioneC: unit -> ('a,'b) relazione
  val addPair: ('a,'b) relazione -> 'a -> 'b -> ('a,'b) relazione
  val removePair: ('a,'b) relazione -> 'a -> 'b -> ('a,'b) relazione
  val isIn: ('a,'b) relazione -> 'a -> 'b -> bool
  val getUno: ('a,'b) relazione -> 'b -> 'a list
```

- Dalla signature si capisce cosa calcola `getUno`?
  – Ovviamente, NO.

API in Java

- **Java** non ha uno specifico meccanismo di API-ADT
- **ADT** emulati in Java, attraverso i meccanismi di:
  - Classe (con generici)
  - Modificatori di accesso di campi e metodi
- **API** emulato in Java, attraverso il meccanismo di:
  - Interfaccia
- Le interfacce di Java forniscono API simili a quelle di OCaml

```
interface RelazioneJAPI<A,B> {
    public RelazioneJAPI<A,B> addPair(A x, B y) throws invalidArgException;
    public RelazioneJAPI<A,B> removePair (A x, B y) throws invalidArgException;
    public boolean isIn(A x, B y)throws invalidArgException;
    public LinkedList<A> getUno(B y) throws invalidArgException;
}
```

- Confrontiamo con la API di Ocaml:
  - Questa API si riferisce a valori IMMUTABLE come quelli di OCaml?
  - Perchè quì sono indicati i nomi dei parametri?
  - Perchè quì è presente il sollevamento di eccezione?
  - Gli additionals devono essere obbligatoriamente dichiarati nell'API?
  - Dalla signature si capisce cosa calcola `getUno`?

Forma del Valore Astratto e Specifica delle Operazioni

- Dalla signature di un'operazione non è possibile capire il comportamento e l'uso dell'operazione.

- API (in OCaml e in Java) deve essere estesa
  - ADT introduce lo Stato Concreto (o Valore Concreto, nei LF)
  - ADT fornisce un'implementazione delle operazioni basata su AF ed I
  - AF è una funzione da Stato Concreto in Valori Astratti:
  - Ma chi deve fornire la Forma (Presentazione) dei Valori Astratti?

- **API** in aggiunta alla signature **deve** fornire:
  - La Forma (Presentazione) dei Valori Astratti
    - ⋆ Questa forma sarà utilizzata poi, negli ADT
  - La specifica delle operazioni introdotte
    - ⋆ trasformazione dello stato e/o Valore Astratto calcolato

API Java estesa con Valore Astratto e Specifica Metodi/1

- Valore Astratto e Convenzioni sulla specifica:
  Immediatamente dopo Intestazione Interfaccia
  - Valore Astratto: La Forma è nella prima linea
  - Convenzioni: Nella seconda linea, notazione `Va(o)`

```
interface RelazioneJAPI<A,B> {
    //Introduce Valori Astratti {(x,y)| x\in[A], y\in[B]}
    //Sia O un oggetto di elazioneJAPI<A,B>, Va(O) \`e il valore astratto definito da O.
    public RelazioneJAPI<A,B> addPair(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola Va tale che:
        // (1) forall(u,v), if this.isIn(u,v) then Va.isIn(u,v)
        // (2) if !this.isIn(u,v) && Va.isIn(u,v) then (u,v)=(x,y)
    public RelazioneJAPI<A,B> removePair (A x, B y) throws invalidArgException;
        //Solleva eccezione ...
        //calcola Va tale che: ....
    public boolean isIn(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola true sse (x,y)\in Va(this)
    public LinkedList<A> getUno(B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola L tale che:
        // this.isIn(x,y)  sse  x=L.get(i) for i\in[0..L.size()-1]
}
```

API Java estesa con Valore Astratto e Specifica Metodi/2

- Specifica Metodo: Immediatamente dopo Intestazione Metodo
  - restrizioni sugli argomenti
  - proprietà della trasformazione di stato e/o del valore calcolato
  - Uso del self-reference this (in agg. agli args)

```
interface RelazioneJAPI<A,B> {
    //Introduce Valori Astratti {(x,y)| x\in[A], y\in[B]}
    //Sia O un oggetto di elazioneJAPI<A,B>, Va(O) \`e il valore astratto definito da O.
    public RelazioneJAPI<A,B> addPair(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola Va tale che:
        // (1) forall(u,v), if this.isIn(u,v) then Va.isIn(u,v)
        // (2) if !this.isIn(u,v) && Va.isIn(u,v) then (u,v)=(x,y)
    public RelazioneJAPI<A,B> removePair (A x, B y) throws invalidArgException;
        //Solleva eccezione ...
        //calcola Va tale che: ....
    public boolean isIn(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola true sse (x,y)\in Va(this)
    public LinkedList<A> getUno(B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola L tale che:
        // this.isIn(x,y)  sse  x=L.get(i) for i\in[0..L.size()-1]
}
```

- Specifica Metodi Additionals: Deve essere inclusa?

Design by Contract in Eiffel

- Tutto è formulato come commento quì
- Linguaggio Eiffel: Formalizzato con *Contracts*
- Linguaggio ADA: Formalizzato con *Axioms*
- Un progetto simile allo studio (da 10 anni) in Java: Assertions

```
interface RelazioneJAPI<A,B> {
    //Introduce Valori Astratti {(x,y)| x\in[A], y\in[B]}
    //Sia O un oggetto di elazioneJAPI<A,B>, Va(O) \`e il valore astratto definito da O.
    public RelazioneJAPI<A,B> addPair(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola Va tale che:
        // (1) forall(u,v), if this.isIn(u,v) then Va.isIn(u,v)
        // (2) if !this.isIn(u,v) && Va.isIn(u,v) then (u,v)=(x,y)
    public RelazioneJAPI<A,B> removePair (A x, B y) throws invalidArgException;
        //Solleva eccezione ...
        //calcola Va tale che: ....
    public boolean isIn(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola true sse (x,y)\in Va(this)
    public LinkedList<A> getUno(B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola L tale che:
        // this.isIn(x,y)  sse  x=L.get(i) for i\in[0..L.size()-1]
}
```

API OCaml estesa con Valore Astratto e Specifica Metodi

- VA e Convenzioni: Imm. dopo Intestazione Interfaccia
- Specifica Metodo: Imm. dopo Intestazione Metodo
- Specifica Metodi Additionals: NON deve essere inclusa.
- Confrontare con quella di Java

```
module type RELAZIONE =
 sig type ('a,'b) relazione
    (* Introduce Valori Astratti \{(x,y)| x\in[A], y\in[B]\} aventi tipo ('a,'b) relazione
    *)
  val relazioneC: unit -> ('a,'b) relazione
    (* calcola relazione vuota \{\}
    *)
  val addPair: ('a,'b) relazione -> 'a -> 'b -> ('a,'b) relazione
    (* forall r,x,y, (addPair r x y)
       calcola relazione Va tale che:
       (1) forall(u,v), if (isIn r u v) then (isIn r u v)
       (2) if !(isIn r u v) && (isIn Va u v) then (u,v)=(x,y)
    *)
  val removePair: ('a,'b) relazione -> 'a -> 'b -> ('a,'b) relazione
    (* ... *)
  val isIn: ('a,'b) relazione -> 'a -> 'b -> bool
    (* forall r,x,y, (isIn r x y)
       calcola relazione true sse (x,y)\in r
    *)
  val getUno: ('a,'b) relazione -> 'b -> 'a list
    (* forall r,y, (getUno r y)
       calcola lista L tale che:
       (isIn r x y)  sse  x=hd(tl^i L) for i\in[0..(length L)-1]
    *)
  val getDue: ('a,'b) relazione -> 'a -> 'b list
    (* ... *)
(* additional *)
  val elements: ('a,'b) relazione -> ('a * 'b) list
  val toString: ('a,'b) relazione -> ('a -> string) -> ('b -> string) -> string
end;;
```

API Java per Valori Mutable estesa con ...

- Notare uso di `thisPre` e `thisPost` nel modificatori
- Occorre distinguere tra il valore di this:
  Prima e Dopo la modifica del suo Stato Concreto

```
interface MuRelazioneAPIX<A,B> {
    //Introduce Valori Astratti {(x,y)| x\in[A], y\in[B]} Mutable
    //Sia O un oggetto di elazioneJAPI<A,B>, Va(O) \`e il valore astratto definito da O.
    public void addPair(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //modifica this in modo tale che: Sia thisPre e thisPost il valore di this prima
        //e dopo l'invocazione, risp.
        // (1) forall(u,v), if thisPres.isIn(u,v) then ThisPost.isIn(u,v)
        // (2) if !thisPre.isIn(u,v) && thisPost.isIn(u,v) then (u,v)=(x,y)
    public void removePair (A x, B y) throws invalidArgException;
        //Solleva eccezione ...
        //calcola Va tale che: ....
    public boolean isIn(A x, B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola true sse (x,y)\in Va(this)
    public LinkedList<A> getUno(B y) throws invalidArgException;
        //Solleva eccezione se (x=null)||(y=null)
        //calcola L tale che:
        // this.isIn(x,y)  sse  x=L.get(i) for i\in[0..L.size()-1]
}
```