

Esercizi su specifiche algebriche

Esercizio 1

- Con riferimento all'algebra ***StringSpec*** vista a lezione, e tenendo conto delle sue proprietà dimostrate negli esempi, dimostrare che, per ogni $s1$, $s2$ e $s3$,
$$\text{append}(s1, \text{append}(s2, s3)) =$$
$$(\text{append}(\text{append}(s1, s2), s3))$$

Esempio StringSpec (*recap*)

algebra StringSpec;

introduces

sorts String, Char, Nat, Bool;

operations

new: () → String;

append: String, String → String;

add: String, Char → String;

length: String → Nat;

isEmpty: String → Bool;

equal: String, String → Bool

Semantica (*recap*)

- Il significato delle operazioni deve essere specificato formalmente.
- La *semantica* di un' algebra è definita da un insieme di *equazioni*
- Rappresentano le *proprietà* essenziali delle operazioni.
- Sono dette anche *assiomi*, in quanto usate per dedurre proprietà delle algebre.

Esempio StringSpec (*recap*)

```
constrains new, append, add, length, isEmpty, equal so that  
String generated by [new, add]  
for all [s, s1, s2: String; c: Char]  
  isEmpty (new ()) = true;  
  isEmpty (add (s, c)) = false;  
  length (new ()) = 0;  
  length (add (s, c)) = length (s) + 1;  
  append (s, new ()) = s; (2)  
  append (s1, add (s2,c)) = add (append (s1,s2),c); (1)  
  equal (new (),new ()) = true;  
  equal (new (), add (s, c)) = false;  
  equal (add (s, c), new ()) = false;  
  equal (add (s1, c), add (s2, c)) = equal (s1,s2);  
end StringSpec.
```

$\text{append}(s1, \text{append}(s2, s3)) =$
 $\text{append}(\text{append}(s1, s2), s3)$

- Per induzione sulla lunghezza di $s3$.
- Per $s3 = \text{new}()$:
- $\text{append}(s1, \text{append}(s2, \text{new}())) =$
 $\text{append}(s1, s2) =$
 $\text{append}(\text{append}(s1, s2), \text{new}())$
- Se la proprietà è vera per ogni $s3$ di lunghezza L : ogni stringa di lunghezza $L+1$ sarà $\text{add}(s3, c)$ per qualche $s3$ e qualche c

$$\text{append}(s1, \text{append}(s2, s3)) = \\ (\text{append}(\text{append}(s1, s2), s3))$$

$$\text{append}(s1, \text{append}(s2, \text{add}(s3, c))) =$$

$$\text{append}(s1, \text{add}(\text{append}(s2, s3), c)) =$$

(assioma)

$$\text{add}(\text{append}(s1, \text{append}(s2, s3)), c) =$$

(assioma)

$$\text{add}(\text{append}(\text{append}(s1, s2), s3), c) =$$

(ipotesi induttiva)

$$\text{append}(\text{append}(s1, s2), \text{add}(s3, c))$$

(assioma)

Esercizio 2

- Scrivere una specifica algebrica del tipo di dato ***Boolean***

Soluzione

algebra BoolAlgebra

introduces

sorts Bool

operations

true: () \rightarrow Bool;

false: () \rightarrow Bool;

not: Bool \rightarrow Bool;

and: Bool x Bool \rightarrow Bool;

or: Bool x Bool \rightarrow Bool;

implies: Bool x Bool \rightarrow Bool;

Soluzione

```
constrains true, false, not, and, or so that  
for all [a,b in Bool]  
not(true()) = false();  
not(false()) = true();  
and(true(),true()) = true();  
and(a, false()) = false();  
and(false(), a) = false();  
or(a,b) = not(and(not(a),not(b)));  
implies(a,b) = or(not(a),b);  
end BoolAlgebra
```

Esercizio 3

Specificare mediante specifiche algebriche il tipo di dato astratto **bag** (insieme con elementi ripetuti), caratterizzato dalle seguenti operazioni:

- *nulbag* per la creazione di un bag vuoto,
- *insert* per l'aggiunta di un elemento,
- *isemptybag* per la verifica di bag vuoto,
- *delete* per la cancellazione di tutte le occorrenze di un elemento,
- *member* per la verifica di presenza di un dato elemento

Soluzione

```
algebra BagAlgebra
imports DataType, BoolAlg
introduces
  sorts Bag;
  operations
    nulbag: () → Bag
    insert: Data x Bag → Bag
    isemptybag: Bag → Bool
    delete: Data x Bag → Bag
    member: Data x Bag → Bool
```

Soluzione

constrains nulbag, insert, isemptybag, delete, member
so that

Bag generated by nulbag, insert

for all [s in Bag, i, e in Data]

isemptybag(nulbag()) = true;

isemptybag(insert(i,s)) = false;

delete(i, nulbag()) = nulbag();

if e = i then delete(e, insert(i,s)) = delete (e,s)

 else delete(e, insert(i,s)) = insert(i, delete(e,s));

member(i, nulbag()) = false;

if e = i then member(e, insert(i,s)) = true

 else member(e, insert(i,s)) = member(e, s);

end BagAlgebra

Esercizio 4

Specificare mediante specifiche algebriche il tipo di dato astratto BAG (insieme con elementi ripetuti), caratterizzato dalle seguenti operazioni:

null, crea il bag vuoto;

is_empty(B), test se bag vuoto;

insert(E,B), inserisce l'elemento E nel bag B;

del_one(E,B), cancella dal bag B la prima occorrenza dell'elemento E, se E appartiene a B, altrimenti lascia B inalterato;

del_all(E,B), cancella dal bag B tutte le copie dell'elemento E (se esistono);

size(B), fornisce il numero di elementi contenuti nel bag B;

howmany(E,B), calcola il numero di elementi uguali a E presenti nel bag B;

project(B), "proietta" il bag B su un insieme S, eliminando da B tutte le copie ripetute di uno stesso elemento.

Soluzione

```
algebra BagAlgebra2
imports DataType, BoolAlg, NatNumb
introduces
  sorts Bag;
  operations
    null: () → Bag;
    is_empty: Bag → Bool;
    insert: Data x Bag → Bag;
    del_one: Data x Bag → Bag;
    del_all: Data x Bag → Bag;
    size: Bag → Nat;
    howmany: Data x Bag → Nat;
    project: Bag → Bag;
```

Soluzione

constrains null, is_empty, insert, del_one, del_all, size, howmany, project **so that**

Bag generated by null, insert

for all [s in Bag, i, e in Data]

is_empty(null()) = true;

is_empty(insert(i,s)) = false;

del_one(i,null()) = null();

if e=i then del_one(e,insert(i,s)) = s

 else del_one(e,insert(i,s)) = insert(i, del_one(e,s));

del_all(i,null()) = null();

if e=i then del_all(e,insert(i,s)) = del_all(e,s)

 else del_all(e,insert(i,s)) = insert(i, del_all(e,s));

size(null()) = 0;

size(insert(i,s)) = 1 + size(s);

Soluzione

```
howmany(e, null()) = 0;
if e=i then howmany(e, insert(i,s)) =
1+howmany(e,s)
else howmany(e,insert(i,s)) = howmany(e,s);
project(null()) = null();
if howmany(i,s) = 0 then project(insert(i,s)) =
        insert(i, project(s))
    else project(insert(i,s)) = project(s);
end BagAlgebra2
```

Esercizio 5

Dare una specifica algebrica del tipo **Pila**, con le seguenti operazioni:

- `new_stack`: creazione di una nuova pila
- `is_empty`: true se la pila è vuota
- `push`: aggiunta di un elemento in cima
- `top`: valore dell'elemento in cima
- `pop`: rimozione dell'elemento in cima

Soluzione

```
algebra StackAlgebra
imports DataType, BoolAlg
introduces
  sorts Stack;
  operations
    new_stack: () → Stack;
    is_empty: Stack → Bool;
    push: Data x Stack → Stack;
    top: Stack → Data;
    pop: Stack → Stack;
```

Soluzione

constrains new_stack, is_empty, push, top, pop **so that**
Stack generated by new_stack, push
for all [i in Data, s in Stack]
 is_empty(new_stack()) = true;
 is_empty(push(i,s)) = false;
 top(new_stack()) = error;
 top(push(i,s)) = i;
 pop(new_stack) = error;
 pop(push(i,s)) = s;
end StackAlgebra

Esercizio 6

- Dimostrare che nell'algebra definita vale la proprietà:

$$\text{push}(i1, \text{push}(i2, s)) = \text{push}(i1, \text{push}(i2, \text{pop}(\text{push}(i3, s))))$$

To do ...