Esercizi su specifiche algebriche

università di ferrara

DA SEICENTO ANNI GUARDIAMO AVANTI.

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,
- append(s1,append(s2,s3))=(append(append(s1,s2),s3))

Soluzione

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

Soluzione

append(s1,append(s2,add(s3,c))) =
append(s1,add(append(s2,s3),c)) =

(assioma)

add(append(s1,append(s2,s3)),c) =

(assioma)

add(append(append(s1,s2),s3),c) =

(ipotesi induttiva)

append(append(s1,s2),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: () $\rightarrow$ Bag
        insert: Data x Bag $\rightarrow$ Bag
        isemptybag: Bag $\rightarrow$ Bool
        delete: Data x Bag $\rightarrow$ Bag
        member: Data x Bag $\rightarrow$ 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 BagAlgebra
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 BagAlgebra

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: () $\rightarrow$ Stack;
        is_empty: Stack $\rightarrow$ Bool;
        push: Data x Stack $\rightarrow$ Stack;
        top: Stack $\rightarrow$ Data;
        pop: Stack $\rightarrow$ 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