

Laborator 3

2015-2016

Programare Logică

Cuprins



1 Specificație pentru liste de numere întregi

2 Specificație pentru arbori binari

Specificație pentru liste de numere întregi

Specificație pentru liste de numere întregi

- Listele se construiesc inductiv plecând de la o listă vidă și adăugând câte un element.
- Avem doi constructori pentru liste:
 - lista vidă (`nil`)
 - adăugarea unui element la o lista.
- Există mai multe variante de a specifica liste.
- În continuare, prezentăm câteva variante.

Varianta 1

```
fmod LIST-INT1 is
  protecting INT .
  sorts NList List .
  subsort NList < List .
  op nil : -> List .
  op _ _ : Int List -> NList .
endfm
```

- Pentru a face diferența între o listă posibil vidă și una nevidă, se consideră două sorturi `NList` și `List`.
- Observați că în această variantă toate listele se termină cu `nil`.

Varianta 2

```
fmod LIST-INT2 is
  protecting INT .
  sorts NList List .
  subsort Int < NList < List .
  op nil : -> List .
  op _ _ : Int List -> NList [id: nil] .
endfm
```

- Deoarece avem subsort Int < NList, numerele întregi sunt văzute ca o listă cu un element.
- Listele se construiesc inductiv fie plecând de la nil, fie de la un întreg.
- Deoarece am adăugat [id: nil], listele nu se mai termină în nil.

Varianta 3

```
fmod LIST-INT3 is
  protecting INT .
  sort List .
  subsort Int < List .
  op nil : -> List .
  op _ _ : List List -> List [assoc id: nil] .
endfm
```

- Listele se obțin prin adăugarea oricărei liste la o altă listă (nu neapărat un singur element adăugat în fața unei liste).
- Din cauza atributului [id: nil], listele nu se termină în nil.

Operații pe liste

- În funcție de ce operații vrem să definim pe liste, alegem o variantă de specificație pentru liste.
- În acest Laborator vom lucra cu **Varianta 2!**
- În următorul modul definim **lungimea** unei liste:

```
fmod LENGTH is
  protecting LIST-INT2 .
  op length : List -> Nat .
  var I : Int .
  var L : List .
  eq length(nil) = 0 .
  eq length(I L) = 1 + length(L) .
endfm
```

```
red length(1 2 3 4 5) .
***> should be 5
```


Exercitiul 1

Plecând de la `LIST-INT2`, definiți următoarele operații:

- 1 Apartenența unui element la o listă

```
op _in_ : Int NList -> Bool
```

- 2 Adăugarea unei liste la o altă listă (concatenare)

```
op append : List List -> List
```

- 3 Inversarea elementelor dintr-o listă

```
op rev : List -> List
```

```
red rev(1 2 3 4 5) .
```

```
***> should be 5 4 3 2 1
```

- 4 Sortarea unei liste

```
op sort : List -> List
```

Exercitiul 2

Plecând de la `LIST-INT3`, definiți operațiile de la Exercițiul 1.

Specificație pentru arbori binari

Arbori binari

Considerăm că un arbore poate fi

- vid sau
- un număr întreg cu un subarbore stâng și unul drept.

```
fmod TREE is
  protecting INT .
  sort Tree .
  op empty : -> Tree .
  op _ _ _ : Tree Int Tree -> Tree .
endfm
```

Operații pe arbori

Următoarea operație întoarce **imaginea în oglindă** a unui arbore

- recursiv înlocuiește fiecare subarbore stâng cu imaginea în oglindă a subarborelui drept și vice-versa.

```
fmod MIRROR is
  protecting TREE .
  op mirror : Tree -> Tree .
  vars L R : Tree . var I : Int .
  eq mirror(empty) = empty .
  eq mirror(L I R) = mirror(R) I mirror(L) .
endfm
```

```
red mirror((empty 3 (empty 1 empty)) 5 ((empty 6 empty) 2 empty)) .
***> should be (empty 2 (empty 6 empty)) 5 ((empty 1 empty) 3 empty)
```

Exercitiul 3

Plecând de la specificația **TREE**, definiți următoarele operații:

- 1 Căutarea în arbori binari

```
op search : Int Tree -> Bool
```

- 2 Adâncimea unui arbore

```
op depth : Tree -> Int
```

- 3 Traversarea în

- 1 inordine op SRD : Tree -> List

- 2 preordine op RSD : Tree -> List

- 3 postordine op SDR : Tree -> List

Exercitiul 4

Scrieți o specificație care folosește arbori binari de căutare pentru a sorta liste de întregi.

- Definiți operația

`bt-insert : NList Tree -> Tree`

care inserează fiecare întreg din listă în locul său în arborele de căutare.

- Definiți operația

`btsort : NList -> NList`

care sortează lista dată ca argument și folosește operația SRD.



Pe săptămâna viitoare!