Laborator 7 - Maude

2016-2017

Programare Logică

Laboratorul 7

TODO

- Liste de numere întregi în Maude.

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 vida (`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 adaugat `[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
```

Practică

Exercițiul 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
   ```

Practică

Exercițiul 2

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

Practică

Exercițiul 3

Scrieți o specificație pentru liste de numere întregi. Constructorul pentru liste trebuie să adauge câte un element la o listă, nu o listă întreagă. În acest modul definiți o relație $<<<$ care compară două liste după regula:

- dacă lungimea listei L1 este strict mai mica decat lungimea listei L2, atunci `L1 <<< L2`;
- dacă lungimea listei L1 este egală cu lungimea listei L2, atunci L1 și L2 se compară element cu element.

Practică

Exercițiul 4

Scrieți o specificație pentru liste de numere întregi. În acest modul definiți:

- `replace(X,Y,L)`: întoarce lista obținută prin înlocuirea fiecărei apariții a lui `X` în `L` cu `Y`.
- `add(X,L)`: întoarce lista obținută prin adaugarea lui `X` pe pozițiile pare din `L`.

Pe săptămâna viitoare!