

Laborator 7

Reminder

Tip de data structurat produs:

Declaratie:

`colset Tip = product Tip1*Tip2*.....*Tipn;`

Unde `Tip1, Tip2, Tipn` sunt tipuri definite anterior

Valoare din acest tip: n-uplu (v_1, v_2, \dots, v_n) , cu v_i valori din `Tipi`

Fie o valoare $v = (v_1, v_2, \dots, v_n)$

Operator specific: `#i`

`# i (v) => vi`

`i = 1, ..., n`

Operatori logici:

`a = b a <> b andalso orelse`

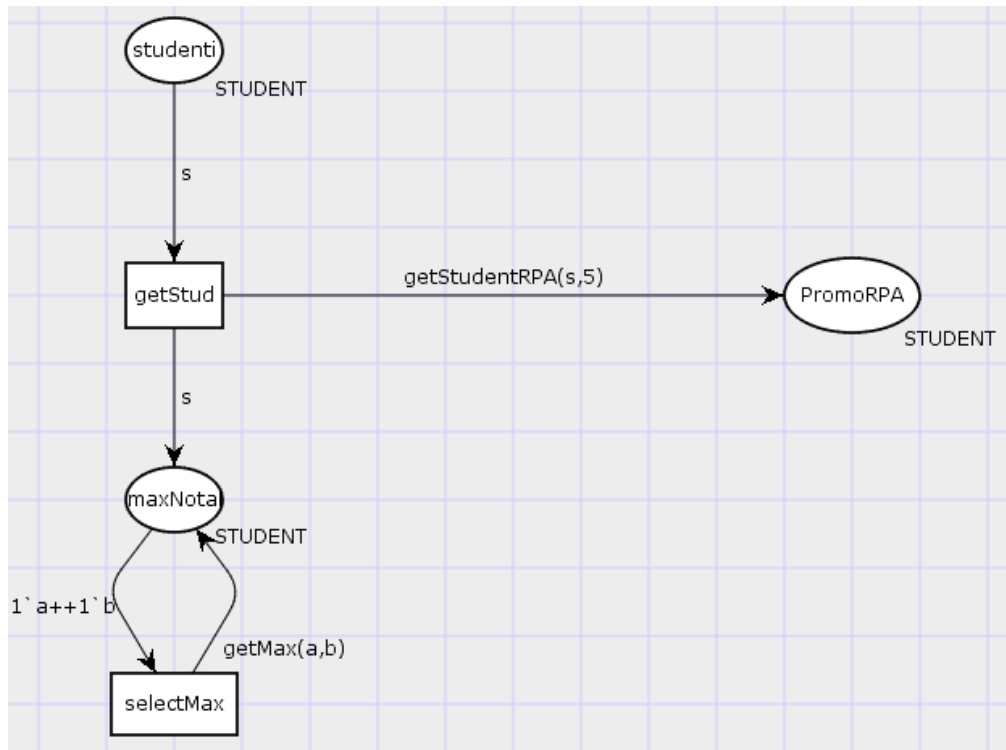
`fun maxNota(...) = empty ;`

Ex1

Fie urmatoarea retea.

```
▼ Declarations
  ► Standard priorities
  ► Standard declarations
  ▼ colset OB = with RPA|SCA|TDN;
  ▼ colset NOTA = int with 0..10;
  ▼ colset STUDENT = product STRING*OB*NOTA;
```

("Gigi", RPA, 10)



- In prima locatie (studenti) completati o marcare initiala (multisetul sa cuprinda macar 6 elemente, cate 2 studenti pentru fiecare optional)
- Completati declaratiile lipsa astfel incat reteaua sa fie corecta
- Scrieti o functie `getStudentRPA(s:STUDENT, nota)`, care returneaza un multiset cu studentul s, daca s este inscris la obiectul RPA si nota este \geq nota, in caz contrar returneaza multisetul vid
- Scrieti o functie `getMax(a:STUDENT, b:STUDENT)` care returneaza un multiset cu studentul care are nota mai mare
- Completati o garda la tranzitia `selectMax`, care sa permita executia tranzitiei doar in cazul in care a si b au note diferite
- Ce marcare va produce o simulare cu ultimul buton din bara de unelte de simulare?

Exemplu de functie care are ca parametru un Student si returneaza un multiset cu studentul, daca acesta nu este promovat sau multisetul vid in caz contrar:

```
fun getStudent(s:STUDENT) = if #3 s < 5 then 1`s else empty;
```

Modificati reteaua de la Ex1: adaugati o locatie "promoSCA" de tip Student, un arc de la `getStud` la noua locatie

Scrieti o functie:

`getStudent(s:STUDENT, f)` (al doilea parametru este o functie), care returneaza un multiset cu studentul daca `f(s)` este true, altfel returneaza multisetul vid

Obs: `getStudent` va fi evaluata corect daca parametrul `f` primit este o functie `f: STUDENT->BOOL`

STUDENT->BOOL

Scrieti apoi o functie `isSCA(s:STUDENT)` care returneaza `true` daca studentul este inscris la obiectul SCA si are nota ≥ 5

Utilizati `getStudent` si `isSCA` pe arcul catre “`promoSCA`” pentru a adauga in noua locatie “`studentiSCA`” un student daca acesta este inscris la SCA si are nota ≥ 5
(expresia de pe arc: `getStudent(s, isSCA)`)

Apoi inlocuiti `getStudentRPA` cu `getStudent(s, isRPA)` (scrieti voi `isRPA`)

Tipul de data List

Daca `Tip` este un tip definit deja,

```
colset LTip = list Tip;
```

defineste un tip lista, avand elemente din `Tip`: $[v_1, v_2, \dots, v_n]$, v_i valori din `Tip`

Se poate defini si:

```
colset LTip = list Tip with n1..n2;
```

(n_1, n_2 sunt limitele pentru lungimea listei)

Lista vida: `[]` sau `nil`

Operatii cu liste (selectie):

`hd l` : daca $l = [v_1, v_2, \dots, v_n]$, returneaza v_1

`tl l` : daca $l = [v_1, v_2, \dots, v_n]$, returneaza lista $[v_2, \dots, v_n]$

`h::t` : reprezinta o lista: daca h este un element si $t = [v_1, \dots, v_n]$ o lista de elemente de acelasi tip cu h , atunci $h::t = [h, v_1, \dots, v_n]$

`h::t` poate fi folosit ca pattern (parametru in functii) sau ca expresie pe arcele din retea

`map f l` : daca $f:Tip1 \rightarrow Tip2$ si $l = [v_1, v_2, \dots, v_n]$ este o lista de elemente de `Tip1`, atunci rezultatul este $[f(v_1), f(v_2), \dots, f(v_n)]$ (o lista `list Tip2`)

Alte functii gasiti aici:

<http://cpntools.org/2018/01/09/list-color-sets/>

Utilizare expresii cu “let”:

```
let
```

```
    val  $v_1$  =  $expr_1$ 
```

```

..... • •
    val vn = exprn
in
    Expr
end

```

Intreaga expresie are tipul expresiei Expr
Expr poate fi orice expresie

Exemplu:

2 + let val x=5 val y=20 in x+y end + 3
se evalueaza la 30

Utilizare definitii clauzale pentru functii

```

fun id pattern1 = expr1
  | id pattern2 = expr2
..... •
  | id patternn = exprn ;

```

Exemplu:

```

fun fact (0) = 1
  | fact(n) = n * fact(n-1);

fun length([]) = 0
  | length(h::t) = 1 + length(t) ;

```

Ex2

Preluati tipurile de date de mai sus

Adaugati:

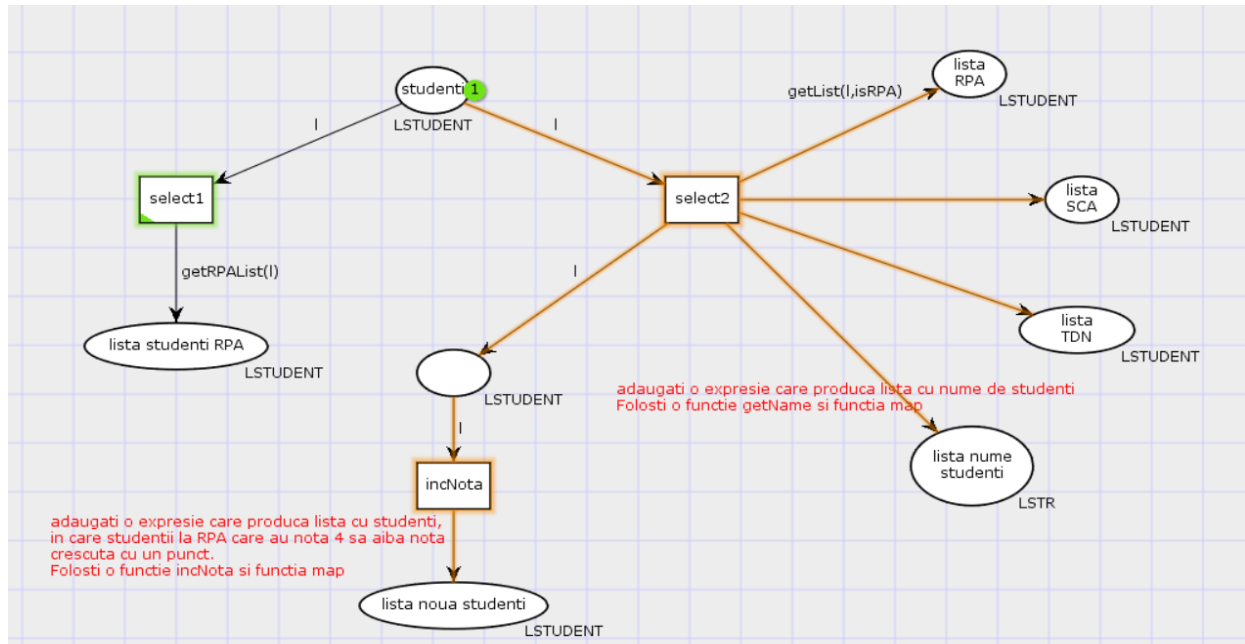
```

colset LSTUDENT = list STUDENT;
colset LSTR = list STRING;

```

Adaugati o marcare initiala in locatia studenti (o lista cu 6 studenti, cate 2 la fiecare optional)

```
1`[(...),(...),(....)]
```



Exemple de functie care returneaza lista studentilor inscrisi la RPA

```
fun getRPAList(l:LSTUDENT) =
  if l=[] then [] else
  if #2 (hd l) = RPA then (hd l)::getRPAList(tl l)
  else getRPAList(tl l);
```

```
fun getRPAList(l:LSTUDENT) =
  if l=[] then [] else
  let
    val h = hd l
    val t = tl l
    val list = getRPAList(t)
  in
    if #2 h = RPA then h::list else list
  end;
```

1. Definiti o noua varianta pentru functia `getRPAList`, care sa fie o definitie clauzala.
Folositi ca pattern-uri: `[]` si `h::t:LSTUDENT`
2. Scrieti o functie generica `getList(l,f)` care se va putea evalua pentru orice lista `l` avand elemente de tip `A` si orice functie `f:A->BOOL` si va produce lista elementelor din `l` pentru care `f` este true (de preferat definitie clauzala)

Scrieti apoi functii `isRPA`, `isSCA` si `isTDN` si folositi `getList` cu aceste functii pentru a obtine lista studentilor inscrisi la RPA, SCA, TDN

3. Adaugati pe arce expresiile lipsa.

Pentru arcul din tranzitia incNota:

Exemplu de functie care creste nota tuturor cu 1 (a se modifica conform cerintei)

```
fun incNota(a:STUDENT) = (#1 a, #2 a, #3 a + 1);
```