Paradigme de Programare

Conf. dr. ing. Andrei Olaru

andrei.olaru@upb.ro
Departamentul de Calculatoare

2025

```
> :t map
map ::  (a -> b) -> [a] -> [b]
```

```
> map ++ 5 tail [1..]
Couldn't match expected type '[a0]'
with actual type '(a1 -> b0) -> [a1] -> [b0]'
In the first argument of '(++)', namely 'map'
In the expression:  map ++ 5 tail [1 ..  ]
In an equation for 'it':  it = map ++ 5 tail [1 ..  ]
```

Cursul 7: Tipuri în Haskell

Tipare

Tipuri

Pentru toate valorile (inclusiv funcții)

- Tipuri ca mulțimi de valori:
  - `Bool = {True, False}`
  - `Natural = {0, 1, 2, ...}`
  - `Char = {'a', 'b', 'c', ...}`
- Rolul tipurilor (vezi cursuri anterioare);
- Tipare statică:
  - etapa de tipare anterioară etapei de evaluare;
  - asocierea fiecărei expresii din program cu un tip;
- Tipare tare: absența conversiilor implicite de tip;
- Expresii de:
  - program: `5, 2 + 3, x && (not y)`
  - tip: `Integer, [Char], Char -> Bool, a`

Tipuri
Exemple de valori

Exemplu

```
1 5                  :: Integer
2 'a'                :: Char
3 (+1)               :: Integer -> Integer
4 [1,2,3]            :: [Integer] -- liste de un singur tip !
5 (True, "Hello")    :: (Bool, [Char])
6 etc.
```

- Tipurile de bază sunt tipurile elementare din limbaj:
  `Bool, Char, Integer, Int, Float, ...`
- Reprezentare uniformă:

```
1   data Integer    =   ... | -2 | -1 | 0 | 1 | 2 | ...
2   data Char       =   'a' | 'b' | 'c' | ...
```

Constructori de tip

⇒ tipuri noi pentru valori sau funcții

- Funcții de tip, ce îmbogățesc tipurile din limbaj.

Ex Constructori de tip predefiniți

```
1  -- Constructorul de tip functie: ->
2  (-> Bool Bool) ⇒ Bool -> Bool
3  (-> Bool (Bool -> Bool)) ⇒ Bool -> (Bool -> Bool)
4
5  -- Constructorul de tip lista: []
6  ([] Bool) ⇒ [Bool]
7  ([] [Bool]) ⇒ [[Bool]]
8
9  -- Constructorul de tip tuplu: (,...,)
10 ((,) Bool Char) ⇒ (Bool, Char)
11 ((,,) Bool ((,) Char [Bool]) Bool)
12            ⇒ (Bool, (Char, [Bool]), Bool)
```

Constructori de tip
Tipurile funcțiilor

- Constructorul `->` este asociativ dreapta:

  `Integer -> Integer -> Integer`

  $\equiv$ `Integer -> (Integer -> Integer)`

Ex Exemplu

```
1  add6          ::  Integer -> Integer -> Integer
2  add6 x y      =   x + y
3
4  f             ::  (Integer -> Integer) -> Integer
5  f g           =   (g 3) + 1
6
7  idd           ::  a -> a          -- functie polimorfica
8  idd x         =   x               -- a: variabila de tip!
```

Sinteză de tip

Sinteza de tip
Definiție

+ | Sinteză de tip – *type inference* – Determinarea automată a tipului unei expresii, pe baza unor reguli precise.

- Adnotările explicite de tip, deși posibile, nenecesare în majoritatea cazurilor
- Dependentă de:
  - componentele expresiei
  - contextul lexical al expresiei
- Reprezentarea tipurilor → expresii de tip:
  - constante de tip: tipuri de bază;
  - variabile de tip: pot fi legate la orice expresii de tip;
  - aplicații ale constructorilor de tip pe expresii de tip.

Proprietăți induse de tipuri

+ | Progres O expresie bine-tipată (căreia i se poate asocia un tip):

- este o valoare (nu este o aplicare de funcție) *sau*
- (este aplicarea unei funcții și) poate fi redusă (vezi β-redex).

+ | Conservare Evaluarea unei expresii bine-tipate produce o expresie bine-tipată – de obicei, cu același tip.

- dacă sinteza de tip pentru expresia E dă tipul t, atunci după reducere, valoarea expresiei E va fi de tipul t.

Exemple de sinteză de tip

Câteva reguli simplificate de sinteză de tip

- Formă: $\dfrac{\texttt{premisa-1 ... premisa-m}}{\texttt{concluzie-1 ... concluzie-n}}$ (nume)

- Funcție: $\dfrac{\texttt{Var :: a} \qquad \texttt{Expr :: b}}{\texttt{\textbackslash Var -> Expr :: a -> b}}$ (TLambda)
- Aplicație: $\dfrac{\texttt{Expr1 :: a -> b} \qquad \texttt{Expr2 :: a}}{\texttt{(Expr1 Expr2) :: b}}$ (TApp)
- Operatorul +: $\dfrac{\texttt{Expr1 :: Int} \qquad \texttt{Expr2 :: Int}}{\texttt{Expr1 + Expr2 :: Int}}$ (T+)
- Literali întregi: $\dfrac{}{\texttt{0, 1, 2, ... :: Int}}$ (TInt)

Exemple de sinteză de tip

Transformare de funcție

Exemplul 1

```
1  f g = (g 3) + 1
```

Exemple de sinteză de tip

Transformare de funcție

Exemplul 1

```
1  f g = (g 3) + 1
```

$$\frac{\texttt{g :: a} \qquad \texttt{(g 3) + 1 :: b}}{\texttt{f :: a -> b}}\ \text{(TLambda)}$$

$$\frac{\texttt{(g 3) :: Int} \qquad \texttt{1 :: Int}}{\texttt{(g 3) + 1 :: Int}}\ \text{(T+)}$$

$$\Rightarrow \texttt{b = Int}$$

$$\frac{\texttt{g :: c -> d} \qquad \texttt{3 :: c}}{\texttt{(g 3) :: d}}\ \text{(TApp)}$$

$$\Rightarrow \texttt{a = c -> d, c = Int, d = Int}$$

$$\Rightarrow \texttt{f :: (Int -> Int) -> Int}$$

Exemple de sinteză de tip

Combinator de punct fix

Exemplul 2

```
1 fix f = f (fix f)
```

Exemple de sinteză de tip

Combinator de punct fix

Exemplul 2

```
1  fix f = f (fix f)
```

$$\frac{\texttt{f :: a} \qquad \texttt{f (fix f) :: b}}{\texttt{fix :: a -> b}}\ (\texttt{TLambda})$$

$$\frac{\texttt{f :: c -> d} \qquad \texttt{(fix f) :: c}}{\texttt{(f (fix f)) :: d}}\ (\texttt{TApp})$$

$$\Rightarrow \texttt{a = c -> d, b = d}$$

$$\frac{\texttt{fix :: e -> g} \qquad \texttt{f :: e}}{\texttt{(fix f) :: g}}\ (\texttt{TApp})$$

$$\Rightarrow \texttt{a -> b = e -> g, a = e, b = g, c = g}$$

$$\Rightarrow \texttt{fix :: (c -> d) -> b = (g -> g) -> g}$$

Exemple de sinteză de tip
O funcție ne-tipabilă

Exemplul 3

```
1  f x = (x x)
```

Exemple de sinteză de tip
O funcție ne-tipabilă

Ex Exemplul 3

```
1  f x = (x x)
```

$$\frac{\texttt{x :: a} \qquad \texttt{(x x) :: b}}{\texttt{f :: a -> b}}\ \texttt{(TLambda)}$$

$$\frac{\texttt{x :: c -> d} \qquad \texttt{x :: c}}{\texttt{(x x) :: d}}\ \texttt{(TApp)}$$

Ecuația `c -> d = c` nu are soluție ($\nexists$ tipuri recursive)
$\Rightarrow$ funcția nu poate fi tipată.

Unificare
Definiție

- la baza sintezei de tip: unificarea → legarea variabilelor în timpul procesului de sinteză, în scopul unificării diverselor formule de tip elaborate.

+ | Unificare Procesul de identificare a valorilor variabilelor din 2 sau mai multe formule, astfel încât substituirea variabilelor prin valorile asociate să conducă la coincidența formulelor.

+ | Substituție O substituție este o mulțime de legări variabilă - valoare.

Unificare
Condiții

- O variabilă de tip `a` unifică cu o expresie de tip `E` doar dacă:
  - `E` $=$ `a` *sau*
  - `E` $\neq$ `a` și `E` nu conține `a` (*occurrence check*).
    Exemplu: `a` unifică cu `b -> c` dar nu cu `a -> b`.
- 2 constante de tip unifică doar dacă sunt egale;
- 2 aplicații de tip unifică doar dacă implică același constructor de tip și argumente ce unifică recursiv.

- Pentru a unifica expresiile de tip:
  - `t1 = (a, [b])`
  - `t2 = (Int, c)`

Ex | Exemplu

- Pentru a unifica expresiile de tip:
  - `t1 = (a, [b])`
  - `t2 = (Int, c)`
- putem avea substituțiile (variante):
  - `S1 = {a ← Int, b ← Int, c ← [Int]}`
  - `S2 = {a ← Int, c ← [b] }`
- Forme comune pentru `S1` respectiv `S2`:
  - `t1/S1 = t2/S1 = (Int, [Int])`
  - `t1/S2 = t2/S2 = (Int, [b])`

+ | ***Most general unifier* – MGU** Cea mai generală substituție sub care formulele unifică. Exemplu: `S2`.

Tip principal
Exemplu și definiție

Ex Exemplu

- Tipurile: `t1 = (a, [b]) , t2 = (Int, c)`
  - MGU: `S = {a ← Int, c ← [b]}`
  - Tipuri mai particulare (instanțe): `(Integer, [Integer])`, `(Integer, [Char])`, etc
- Funcția: `\x -> x`
  - Tipuri corecte: `Int -> Int , Bool -> Bool , a -> a`

+ Tip principal al unei expresii – Cel mai general tip care descrie complet natura expresiei. Se obține prin utilizarea MGU.

TDA

Constructorul de tip `Natural`

Exemplu de definire TDA 1

Exemplu

```
1  data Natural     =   Zero
2                   | Succ Natural
3      deriving (Show, Eq)
4
5  unu                  =   Succ Zero
6  doi                  =   Succ unu
7
8  addNat Zero n        =   n
9  addNat (Succ m) n    =   Succ (addNat m n)
```

Constructorul de tip `Natural`

Comentarii

- Constructor de tip: `Natural`
  - nular;
  - se confundă cu tipul pe care-l construiește.
- Constructori de date:
  - `Zero`: nular
  - `Succ`: unar
- Constructorii de date ca funcții, dar utilizabile în *pattern matching*.

```
1 Zero :: Natural
2 Succ :: Natural -> Natural
```

Constructorul de tip `Pair`

Exemplu de definire TDA 2

Exemplu

```
1  data Pair a b    = P a b
2      deriving (Show, Eq)
3
4  pair1            =   P 2 True
5  pair2            =   P 1 pair1
6
7  myFst (P x y)    =   x
8  mySnd (P x y)    =   y
```

Constructorul de tip `Pair`

Comentarii

- Constructor de tip: `Pair`
  - polimorfic, binar;
  - generează un tip în momentul aplicării asupra 2 tipuri.
- Constructor de date: `P`, binar:

```
1 P :: a -> b -> Pair a b
```

Sfârșitul cursului 7
Elemente esențiale

- tipuri în Haskell
- expresii de tip și construcție de tipuri
- sinteză de tip, unificare

+ Dați feedback la acest curs aici:
[https://forms.gle/Q4GP6YMRzCV657cRA]