Paradigme de Programare

Ș.l. dr. ing. Andrei Olaru

Departamentul Calculatoare

2015 – 2016, semestrul 2

Cursul 3: Anexă: TDA pentru calcul λ

Cursul 3: Anexă: TDA pentru calcul λ

1. Introducere
2. Lambda-expresii
3. Reducere
4. Evaluare
5. Limbajul lambda-0 și incursiune în TDA
1. TDA

TDA

Notație

$\rightarrow$ Folosim notația:

- $\lambda x_1.\lambda x_2.\ldots.\lambda x_n.E \rightarrow \lambda x_1 x_2 \ldots x_n.E$
- $((\ldots((E\ A_1)\ A_2)\ \ldots)\ A_n) \rightarrow (E\ A_1\ A_2\ \ldots\ A_n)$

Elemente TDA

→ Pentru definirea unui Tip de Date Abstract avem nevoie de:

- Constructori de bază → un set minimal de funcții, prin aplicarea (eventual, repetată) cărora se poate construi oricare element din mulțimea de valori a tipului.
  - constructorii de bază construiesc valorile.
- Operatori → setul complet de funcții care pot lucra cu valorile din tipul de bază.
  - operatorii arată ce operații putem face cu valorile.
- Axiome → definesc rezultatul operatorilor pentru toate posibilele valori (ne ajutăm de constructorii de bază).
  - axiomele arată ce rezultate obținem din operații.

TDA *Bool*
Specificare

· Constructori:

$$\begin{array}{l|ll} T: & \to Bool \\ F: & \to Bool \end{array}$$

· Operatori:

$$\begin{array}{l|ll} not: & Bool \to Bool \\ and: & Bool^2 \to Bool \\ or: & Bool^2 \to Natural \end{array}$$

· Axiome:

$$\begin{array}{l|ll} not: & not(T) = F \\ & not(F) = T \\ and: & and(T, a) = a \\ & and(F, a) = F \\ or: & or(T, a) = T \\ & or(F, a) = a \end{array}$$

TDA *Bool*
Implementarea constructorilor de bază

Intuiție: selecția între cele două valori, *true* și *false*

- $T \equiv_{\text{def}} \lambda x\ y.x$
- $F \equiv_{\text{def}} \lambda x\ y.y$
- Comportament de selectori:
  - $(T\ a\ b) \to (\lambda x\ y.x\ a\ b) \to a$
  - $(F\ a\ b) \to (\lambda x\ y.y\ a\ b) \to b$

TDA *Bool*
Implementarea operatorilor

- $not \equiv_{\text{def}} \lambda x.(x\ F\ T)$
  - $(not\ T) \to (\lambda x.(x\ F\ T)\ T) \to (T\ F\ T) \to F$
  - $(not\ F) \to (\lambda x.(x\ F\ T)\ F) \to (F\ F\ T) \to T$

- $and \equiv_{\text{def}} \lambda x\ y.(x\ y\ F)$
  - $(and\ T\ a) \to (\lambda x\ y.(x\ y\ F)\ T\ a) \to (T\ a\ F) \to a$
  - $(and\ F\ a) \to (\lambda x\ y.(x\ y\ F)\ F\ a) \to (F\ a\ F) \to F$

- $or \equiv_{\text{def}} \lambda x\ y.(x\ T\ y)$
  - $(or\ T\ a) \to (\lambda x\ y.(x\ T\ y)\ T\ a) \to (T\ T\ a) \to T$
  - $(or\ F\ a) \to (\lambda x\ y.(x\ T\ y)\ F\ a) \to (F\ T\ a) \to a$

TDA *Bool*
Condiționala *if*

· Operator: $\mid$ $if : Bool \times A \times A \to A$

· Axiome: $\mid$ $if(T, a, b) = a$
$if(F, a, b) = b$

- Implementare: $if \equiv_{\text{def}} \lambda c\ t\ e.(c\ t\ e)$
  - $(if\ T\ a\ b) \to (\lambda c\ t\ e.(c\ t\ e)\ T\ a\ b) \to (T\ a\ b) \to a$
  - $(if\ F\ a\ b) \to (\lambda c\ t\ e.(c\ t\ e)\ F\ a\ b) \to (F\ a\ b) \to b$

- Funcție nestrictă!

TDA *Pair*
Specificare

- Constructori: $\mid \; pair : A \times B \to Pair$
- Operatori: $\mid \; fst : Pair \to A$, $snd : Pair \to B$
- Axiome: $\mid \; snd(pair(a, b)) = b$, $fst(pair(a, b)) = a$

TDA *Pair*
Implementare

Intuiție: pereche $\rightarrow$funcție ce așteaptă selectorul, pentru a-l aplica asupra membrilor

- $pair \equiv_{\text{def}} \lambda x\ y\ z.(z\ x\ y)$
  - $(pair\ a\ b) \rightarrow (\lambda x\ y\ z.(z\ x\ y)\ a\ b) \rightarrow \lambda z.(z\ a\ b)$
- $fst \equiv_{\text{def}} \lambda p.(p\ T)$
  - $(fst\ (pair\ a\ b)) \rightarrow (\lambda p.(p\ T)\ \lambda z.(z\ a\ b)) \rightarrow (\lambda z.(z\ a\ b)\ T) \rightarrow (T\ a\ b) \rightarrow a$
- $snd \equiv_{\text{def}} \lambda p.(p\ F)$
  - $(snd\ (pair\ a\ b)) \rightarrow (\lambda p.(p\ F)\ \lambda z.(z\ a\ b)) \rightarrow (\lambda z.(z\ a\ b)\ F) \rightarrow (F\ a\ b) \rightarrow b$

TDA *List*
Specificare

- Constructori:

$$\begin{array}{ll} nil: & \to List \\ cons: & A \times List \to List \end{array}$$

- Operatori:

$$\begin{array}{ll} car: & List \setminus \{nil\} \to A \\ cdr: & List \setminus \{nil\} \to List \\ null?: & List \to Bool \\ append: & List^2 \to List \end{array}$$

- Axiome:

$$\begin{array}{ll} car: & car(cons(e, L)) = e \\ cdr: & cdr(cons(e, L)) = L \\ null?: & null?(nil) = T \\ & null?(cons(e, L)) = F \\ append: & append(nil, B) = B \\ & append(cons(e, A), B) = cons(e, append(A, B)) \end{array}$$

TDA *List*
Implementare

Intuiție: listă $\rightarrow$ pereche (*head*, *tail*)

- $nil \equiv_{\text{def}} \lambda x.T$
- $cons \equiv_{\text{def}} pair$
  - $(cons\ e\ L) \rightarrow (\lambda x\ y\ z.(z\ x\ y)\ e\ L) \rightarrow \lambda z.(z\ e\ L)$
- $car \equiv_{\text{def}} fst$
- $cdr \equiv_{\text{def}} snd$
- $null? \equiv_{\text{def}} \lambda L.(L\ \ \lambda x\ y.F)$
  - $(null?\ nil) \rightarrow (\lambda L.(L\ \ \lambda x\ y.F)\ \ \lambda x.T) \rightarrow (\lambda x.T\ \ \ldots) \rightarrow T$
  - $(null?\ (cons\ e\ L)) \rightarrow (\lambda L.(L\ \ \lambda x\ y.F)\ \ \lambda z.(z\ e\ L)) \rightarrow$
    $(\lambda z.(z\ e\ L)\ \ \lambda x\ y.F) \rightarrow (\lambda x\ y.F\ e\ L) \rightarrow F$

TDA *List*
Implementare *append*

- $append \equiv_{\text{def}}$
  $\lambda A\, B.(if\ (null?\ A)\ B\ (cons\ (car\ A)\ (append\ (cdr\ A)\ B)))$

· Problemă: expresia nu admite formă închisă! $\rightarrow$ vezi eliminarea recursivității textuale.

TDA *Natural*
Specificare

· Constructori:

$$\begin{array}{l|ll} & zero: & \to Natural \\ & succ: & Natural \to Natural \end{array}$$

· Operatori:

$$\begin{array}{l|ll} & pred: & Natural \setminus \{zero\} \to Natural \\ & zero?: & Natural \to Bool \\ & add: & Natural^2 \to Natural \end{array}$$

· Axiome:

$$\begin{array}{l|ll} & pred: & pred(succ(n)) = n \\ & zero?: & zero?(zero) = T \\ & & zero?(succ(n)) = F \\ & add: & add(zero, n) = n \\ & & add(succ(n), m) = succ(add(n, m)) \end{array}$$

TDA *Natural*
Implementare

Intuiție: număr $\rightarrow$ listă cu lungimea egală cu valoarea numărului

- $zero \equiv_{\text{def}} nil$
- $succ \equiv_{\text{def}} \lambda n.(cons\ nil\ n)$
- $pred \equiv_{\text{def}} cdr$
- $zero? \equiv_{\text{def}} null$
- $add \equiv_{\text{def}} append$