Fundamentele limbajelor de programare

C04

Denisa Diaconescu
Traian Șerbănuță

Departamentul de Informatică, FMI, UB

Expresivitatea λ-calculului

Expresivitatea λ-calculului

Deși lambda calculul constă doar în λ-termeni, putem reprezenta și manipula tipuri de date comune.

Vom vedea cum putem reprezenta:

- valori booleene
- numere naturale

Booleeni

Booleeni

Vrem să definim λ-termeni care să reprezinte constantele booleene.

Sunt mai multe modalități, una dintre ele fiind:

- $\mathbf{T} \triangleq \lambda xy.x$ (dintre cele două alternative o alege pe prima)
- $\mathbf{F} \triangleq \lambda xy.y$ (dintre cele două alternative o alege pe a doua)

Booleeni

$$\mathbf{T} \triangleq \lambda xy.x \qquad \mathbf{F} \triangleq \lambda xy.y$$

Acum am vrea să definim un test condiționat **if**.

Am vrea ca **if** să ia trei argumente b, t, f, unde b este o valoare booleană, iar t, f sunt λ-termeni oarecare.

Funcția ar trebui să returneze t dacă $b = true$ și f dacă $b = false$

$$\mathbf{if} = \lambda btf. \begin{cases} t, & \text{if } b = true, \\ f, & \text{if } b = false. \end{cases}$$

Deoarece $\mathbf{T}\ t\ f \twoheadrightarrow_\beta t$ și $\mathbf{F}\ t\ f \twoheadrightarrow_\beta f$, **if** trebuie doar să aplice argumentul său boolean celorlalte argumente:

$$\mathbf{if} \triangleq \lambda btf.b\ t\ f$$

Booleni

$$\mathbf{T} \triangleq \lambda xy.x \qquad \mathbf{F} \triangleq \lambda xy.y \qquad \mathbf{if} \triangleq \lambda btf.b\ t\ f$$

Celelalte operații booleene pot fi definite folosind **if**:

$$\begin{aligned}
\mathbf{and} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ b_2\ \mathbf{F} \\
\mathbf{or} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ \mathbf{T}\ b_2 \\
\mathbf{not} &\triangleq \lambda b_1.\mathbf{if}\ b_1\ \mathbf{F}\ \mathbf{T}
\end{aligned}$$

Observați că aceste operații lucrează corect doar dacă primesc ca argumente valori booleene.

Nu există nicio garanție să se comporte rezonabil pe orice alți λ-termeni.

Folosind lambda calcul fără tipuri, avem *garbage in, garbage out*.

Codările nu sunt unice. De exemplu, pentru **and** am fi putut folosi codarile $\lambda b_1 b_2.b_2\ b_1\ b_2$ sau $\lambda b_1 b_2.b_1\ b_2\ \mathbf{F}$.

Booleni

$$\mathbf{T} \triangleq \lambda xy.x \qquad \mathbf{F} \triangleq \lambda xy.y \qquad \mathbf{if} \triangleq \lambda btf.b\ t\ f$$

$$\begin{aligned} \mathbf{and} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ b_2\ \mathbf{F} \\ \mathbf{or} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ \mathbf{T}\ b_2 \\ \mathbf{not} &\triangleq \lambda b_1.\mathbf{if}\ b_1\ \mathbf{F}\ \mathbf{T} \end{aligned}$$

Exercițiu. Aduceți la o formă normală următorii termenii:

- **and T F**
- **or F T**
- **not T**

Booleeni

$$\mathbf{T} \triangleq \lambda xy.x \qquad \mathbf{F} \triangleq \lambda xy.y \qquad \mathbf{if} \triangleq \lambda btf.b\ t\ f$$

$$\begin{aligned} \mathbf{and} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ b_2\ \mathbf{F} \\ \mathbf{or} &\triangleq \lambda b_1 b_2.\mathbf{if}\ b_1\ \mathbf{T}\ b_2 \\ \mathbf{not} &\triangleq \lambda b_1.\mathbf{if}\ b_1\ \mathbf{F}\ \mathbf{T} \end{aligned}$$

Soluții:

$$\begin{aligned} \mathbf{and}\,\mathbf{T}\mathbf{F} &= (\lambda b_1 b_2.\mathbf{if}\ b_1\ b_2\ \mathbf{F})\,\mathbf{T}\mathbf{F} \twoheadrightarrow_\beta \mathbf{if}\ \mathbf{T}\ \mathbf{F}\ \mathbf{F} = (\lambda btf.b\ t\ f)\ \mathbf{T}\ \mathbf{F}\ \mathbf{F} \\ &\twoheadrightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{F} = (\lambda xy.x)\ \mathbf{F}\ \mathbf{F} \twoheadrightarrow_\beta \mathbf{F} \end{aligned}$$

$$\begin{aligned} \mathbf{or}\,\mathbf{F}\mathbf{T} &= (\lambda b_1 b_2.\mathbf{if}\ b_1\ \mathbf{T}\ b_2)\,\mathbf{F}\mathbf{T} \twoheadrightarrow_\beta \mathbf{if}\ \mathbf{F}\ \mathbf{T}\ \mathbf{T} = (\lambda btf.b\ t\ f)\ \mathbf{F}\ \mathbf{T}\ \mathbf{T} \\ &\twoheadrightarrow_\beta \mathbf{F}\ \mathbf{T}\ \mathbf{T} = (\lambda xy.y)\ \mathbf{T}\ \mathbf{T} \twoheadrightarrow_\beta \mathbf{T} \end{aligned}$$

$$\begin{aligned} \mathbf{not}\,\mathbf{T} &= (\lambda b_1.\mathbf{if}\ b_1\ \mathbf{F}\ \mathbf{T})\,\mathbf{T} \twoheadrightarrow_\beta \mathbf{if}\ \mathbf{T}\ \mathbf{F}\ \mathbf{T} = (\lambda btf.b\ t\ f)\ \mathbf{T}\ \mathbf{F}\ \mathbf{T} \\ &\twoheadrightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{T} = (\lambda xy.x)\ \mathbf{F}\ \mathbf{T} \twoheadrightarrow_\beta \mathbf{F} \end{aligned}$$

Numere naturale

Numere naturale

Vom reprezenta numerele naturale $\mathbb{N}$ folosind numeralii Church.

Numeralul Church pentru numărul $n \in \mathbb{N}$ este notat $\overline{n}$.

Numeralul Church $\overline{n}$ este λ-termenul $\lambda fx.f^n\, x$, unde f^n reprezintă compunerea lui f cu ea însăși de n ori:

$$
\begin{array}{lclcl}
\overline{0} & \triangleq & \lambda fx.f^0\, x & = & \lambda fx.x \\
\overline{1} & \triangleq & \lambda fx.f^1\, x & = & \lambda fx.f\, x \\
\overline{2} & \triangleq & \lambda fx.f^2\, x & = & \lambda fx.f\,(f\, x) \\
\overline{3} & \triangleq & \lambda fx.f^3\, x & = & \lambda fx.f\,(f\,(f\, x)) \\
 & \vdots & & & \\
\overline{n} & \triangleq & \lambda fx.f^n\, x & = & \lambda fx.\underbrace{f(f(\ldots(f}_{n}\ x)\ldots))
\end{array}
$$

Numere naturale

$$\overline{n} \triangleq \lambda fx.f^n\, x$$

Acum putem defini funcția succesor prin

$$\textbf{Succ} \triangleq \lambda nfx.f\,(n\,f\,x)$$

Observați că **Succ** pe argumentul $\overline{n}$ returnează o funcție care primește ca argument o funcție f, îi aplică $\overline{n}$ pentru a obține compunerea de n ori a lui f cu ea însăși, apoi aplică iar f pentru a obține compunerea de $n+1$ ori a lui f cu ea însăși.

$$\begin{array}{rcl}
\textbf{Succ}\,\overline{n} & = & (\lambda nfx.f\,(n\,f\,x))\,\overline{n} \\
& \twoheadrightarrow_\beta & \lambda fx.f\,(\overline{n}\,f\,x) \\
& \twoheadrightarrow_\beta & \lambda fx.f\,(f^n\,x) \\
& = & \lambda fx.f^{n+1}\,x \\
& = & \overline{n+1}
\end{array}$$

Numere naturale

$$\overline{n} \triangleq \lambda fx.f^n\, x \qquad \textbf{Succ} \triangleq \lambda nfx.f\,(n\,f\,x)$$

Putem face operații aritmetice de bază cu numeralii Church.

Pentru adunare, putem defini

$$\textbf{add} \triangleq \lambda mnfx.m\,f\,(n\,f\,x)$$

Pentru argumentele $\overline{m}$ și $\overline{n}$, obținem:

$$\begin{aligned}
\textbf{add}\,\overline{m}\,\overline{n} &= (\lambda mnfx.m\,f\,(n\,f\,x))\,\overline{m}\,\overline{n} \\
&\twoheadrightarrow_\beta \lambda fx.\overline{m}\,f\,(\overline{n}\,f\,x) \\
&\twoheadrightarrow_\beta \lambda fx.f^m\,(f^n\,x) \\
&= \lambda fx.f^{m+n}\,x \\
&= \overline{m+n}
\end{aligned}$$

Am folosit compunerea lui f^m cu f^n pentru a obține f^{m+n}.

$$\overline{n} \triangleq \lambda fx.f^n\,x \qquad \textbf{Succ} \triangleq \lambda nfx.f\,(n\,f\,x)$$

Putem defini adunarea și ca aplicarea repetată a funcției succesor:

$$\textbf{add}' \triangleq \lambda mn.m\,\textbf{Succ}\,n$$

$$\begin{aligned}
\textbf{add}'\,\overline{m}\,\overline{n} &= (\lambda mn.m\,\textbf{Succ}\,n)\,\overline{m}\,\overline{n} \\
&\twoheadrightarrow_\beta \overline{m}\,\textbf{Succ}\,\overline{n} \\
&= (\lambda fx.f^m\,x)\,\textbf{Succ}\,\overline{n} \\
&\twoheadrightarrow_\beta \textbf{Succ}^m\,\overline{n} \\
&= \underbrace{\textbf{Succ}(\textbf{Succ}(\ldots(\textbf{Succ}}_{m}\,\overline{n})\ldots)) \\
&\twoheadrightarrow_\beta \underbrace{\textbf{Succ}(\textbf{Succ}(\ldots(\textbf{Succ}}_{m-1}\,\overline{n+1})\ldots)) \\
&\twoheadrightarrow_\beta \overline{m+n}
\end{aligned}$$

Numere naturale

$$\overline{n} \triangleq \lambda fx.f^n\, x \qquad\qquad \textbf{Succ} \triangleq \lambda nfx.f\,(n\,f\,x)$$

$$\textbf{add}' \triangleq \lambda mn.m\,\textbf{Succ}\,n$$

Similar înmulțirea este adunare repetată, iar ridicarea la putere este înmulțire repetată:

$$\begin{aligned} \textbf{mul} &\triangleq \lambda mn.m\,(\textbf{add}\,n)\,\overline{0} \\ \textbf{exp} &\triangleq \lambda mn.m\,(\textbf{mul}\,n)\,\overline{1} \end{aligned}$$

Numere naturale

Putem defini o funcție de la numere naturale la booleeni care verifică dacă un număr natural este 0 sau nu

$$\begin{aligned} \textbf{isZero}(0) &= \textit{true} \\ \textbf{isZero}(n) &= \textit{false} \qquad \text{dacă } n \neq 0 \end{aligned}$$

O codare în lambda calcul a unei astfel de funcții este

$$\textbf{isZero} \triangleq \lambda nxy.n\,(\lambda z.y)\,x$$

Exercițiu. Verificați afirmația de mai sus.

Putem să definim și codarea **pred** pentru predecesorul unui număr natural. Această codare nu este deloc ușoară și alegem să lucrăm cu ea ca cu o cutie neagră.

Putem exprima mai mult?

Avem văzut codari simple pentru booleeni și numere naturale.

Totuși nu avem o metodă pentru a construi astfel de λ-termeni.

Ne trebuie un mecanism care să ne permită să construim funcții mai complicate din funcții mai simple.

De exemplu, să considerăm funcția factorial

$$
\begin{array}{lcll}
0! & = & 1 & \\
n! & = & n \cdot (n-1)!, & \text{dacă } n \neq 0
\end{array}
$$

Puncte fixe

Puncte fixe

Fie f o funcție. Spunem că x este un punct fix al lui f dacă f(x) = x.

În matematică, unele funcții au puncte fixe, altele nu au.

De exemplu, $f(x) = x^2$ are două puncte fixe 0 și 1, dar $f(x) = x + 1$ nu are puncte fixe.

Mai mult, unele funcții au o infinitate de puncte fixe, cum ar fi $f(x) = x$.

β-echivalență

Am notat cu $M \twoheadrightarrow_\beta M'$ faptul că M poate fi β-redus până la M' în 0 sau mai mulți pași de β-reducție.

$\twoheadrightarrow_\beta$ este închiderea reflexivă și tranzitivă a relației $\rightarrow_\beta$.

Notăm cu $M =_\beta M'$ faptul că M poate fi transformat în M' în 0 sau mai mulți pași de β-reducție, transformare în care pașii de reducție pot fi și întorși.

$=_\beta$ este închiderea reflexivă, simetrică și tranzitivă a relației $\rightarrow_\beta$.

De exemplu, avem $(\lambda y.y\, v)\, z =_\beta (\lambda x.z\, x)\, v$ deoarece avem

$$(\lambda y.y\, v)\, z \rightarrow_\beta z\, v \leftarrow_\beta (\lambda x.z\, x)\, v$$

Notăm cu $\leftarrow_\beta$ inversul relației $\rightarrow_\beta$.

Puncte fixe în lambda-calcul

Dacă F și M sunt λ-termeni, spunem că M este un punct fix al lui F dacă $F\,M =_\beta M$.

Thm. În lambda calculul fără tipuri, orice termen are un punct fix.

Puncte fixe în lambda-calcul

Dacă F și M sunt λ-termeni, spunem că M este un punct fix al lui F dacă $F\,M =_\beta M$.

Thm. În lambda calculul fără tipuri, orice termen are un punct fix.

Dem. Vrem să arătăm că pentru orice termen F există un termen M astfel încât $F\,M =_\beta M$.

Fie F un termen. Considerăm $M \triangleq (\lambda x.F\,(x\,x))\,(\lambda x.F\,(x\,x))$. Avem

$$\begin{array}{rcl} M & = & (\lambda x.F\,(x\,x))\,(\lambda x.F\,(x\,x)) \\ & \to_\beta & F\,((\lambda x.F\,(x\,x))\,(\lambda x.F\,(x\,x))) \\ & = & F\,M \end{array}$$

Deci avem $FM =_\beta M$.

Combinatori de punct fix

Combinatorii de puncte fixe sunt termeni închiși care "construiesc" un punct fix pentru un termen arbitrar.

Câteva exemple:

- Combinatorul de punct fix al lui Curry
  $\mathbf{Y} \triangleq \lambda y.(\lambda x.y\,(x\,x))\,(\lambda x.y\,(x\,x))$
  Pentru orice termen F, $\mathbf{Y}F$ este un punct fix al lui F deoarece $\mathbf{Y}F \twoheadrightarrow_\beta F\,(\mathbf{Y}\,F)$.
- Combinatorul de punct fix al lui Turing
  $\Theta \triangleq (\lambda xy.y\,(x\,x\,y))\,(\lambda xy.y\,(x\,x\,y))$
  Pentru orice termen F, ΘF este un punct fix al lui F deoarece $\Theta F \twoheadrightarrow_\beta F\,(\Theta\,F)$.

Rezolvarea de ecuații în lambda calcul

Punctele fixe ne permit să rezolvăm ecuații. A găsi un punct fix pentru f este același lucru cu a rezolva o ecuație de forma

$$x = f(x)$$

Am văzut că în lambda calcul există mereu o soluție pentru astfel de ecuații.

Rezolvarea de ecuații în lambda calcul

Să aplicăm această idee pentru funcția factorial.

Cea mai naturală definiție a funcției factorial este cea recursivă și o putem scrie în lambda calcul prin

$$\mathbf{fact}\, n = \mathbf{if}\,(\mathbf{isZero}\, n)\,(\overline{1})\,(\mathbf{mul}\, n\,(\mathbf{fact}(\mathbf{pred}\, n))\,)$$

În ecuația de mai sus, **fact** apare și în stânga, și în dreapta. Pentru a găsi cine este **fact**, trebuie să rezolvăm o ecuație.

Rezolvarea de ecuații în lambda calcul

Să rezolvăm ecuația de mai sus. Rescriem problema puțin

$$\begin{aligned}\mathbf{fact} &= \lambda n.\mathbf{if}\,(\mathbf{isZero}\,n)\,(\overline{1})\,(\mathbf{mul}\,n\,(\mathbf{fact}(\mathbf{pred}\,n))\,)\\ \mathbf{fact} &= (\lambda fn.\mathbf{if}\,(\mathbf{isZero}\,n)\,(\overline{1})\,(\mathbf{mul}\,n\,(f(\mathbf{pred}\,n))\,))\,\mathbf{fact}\end{aligned}$$

Notăm termenul $\lambda fn.\mathbf{if}\,(\mathbf{isZero}\,n)\,(\overline{1})\,(\mathbf{mul}\,n\,(f(\mathbf{pred}\,n))\,)$ cu F.

Ultima ecuație devine $\mathbf{fact} = F\,\mathbf{fact}$, o ecuație de punct fix.

Am văzut că $\mathbf{Y}\,F$ este un punct fix pentru F (adică $\mathbf{Y}\,F \twoheadrightarrow_\beta F\,(\mathbf{Y}\,F)$), de aceea putem rezolva ecuația de mai sus luând

$$\begin{aligned}\mathbf{fact} &\triangleq \mathbf{Y}\,F\\ \mathbf{fact} &\triangleq \mathbf{Y}\,(\lambda fn.\mathbf{if}\,(\mathbf{isZero}\,n)\,(\overline{1})\,(\mathbf{mul}\,n\,(f(\mathbf{pred}\,n))\,))\end{aligned}$$

Observați că **fact** a dispărut din partea dreaptă.

Exercițiu. Evaluați $\mathbf{fact}\,\overline{2}$ ținând cont că $\mathbf{fact} \twoheadrightarrow_\beta F\,\mathbf{fact}$.

Quiz time!

https://tinyurl.com/C04-Quiz1

Pe săptămâna viitoare!