Fundamentele limbajelor de programare

C03

Denisa Diaconescu
Traian Șerbănuță

Departamentul de Informatică, FMI, UB

Lambda calcul - β-reducții

β-reducții

Convenție. Spunem că doi termeni sunt egali, notat $M = N$, dacă sunt α-echivalenți.

- β-reducție = procesul de a evalua lambda termeni prin "pasarea de argumente funcțiilor"
- β-redex = un termen de forma $(\lambda x.M)\, N$
- redusul unui redex $(\lambda x.M)\, N$ este $M[N/x]$
- reducem lambda termeni prin găsirea unui subtermen care este redex, și apoi înlocuirea acelui redex cu redusul său
- repetăm acest proces de câte ori putem, până nu mai sunt redex-uri
- formă normală = un lambda termen fără redex-uri

β-reducții

Un pas de β-reducție $\rightarrow_\beta$ este cea mai mică relație pe lambda termeni care satisface regulile:

$$(\beta) \quad \frac{}{(\lambda x.M)N \rightarrow_\beta M[N/x]}$$

$$(cong_1) \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N}$$

$$(cong_2) \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

$$(\xi) \quad \frac{M \rightarrow_\beta M'}{\lambda x.M \rightarrow_\beta \lambda x.M'}$$

β-reducții

La fiecare pas, subliniem redexul ales în procesul de β-reducție.

$$
\begin{array}{rcl}
(\lambda x.y)\,(\underline{(\lambda z.zz\,)\,(\lambda w.w)}) & \to_\beta & (\lambda x.y)\,((z\,z)[\lambda w.w/z]) \\
& \equiv & (\lambda x.y)\,((z[\lambda w.w/z])\,(z[\lambda w.w/z]) \\
& \equiv & (\lambda x.y)\,(\underline{(\lambda w.w)\,(\lambda w.w)}) \\
& \to_\beta & \underline{(\lambda x.y)\,(\lambda w.w)} \\
& \to_\beta & y
\end{array}
$$

Ultimul termen nu mai are redex-uri, deci este în formă normală.

β-reducții

$$
\begin{aligned}
(\lambda x.y)\,(\underline{(\lambda z.zz)\,(\lambda w.w)}) &\to_\beta (\lambda x.y)\,(\underline{(\lambda w.w)\,(\lambda w.w)}) \\
&\to_\beta \underline{(\lambda x.y)\,(\lambda w.w)} \\
&\to_\beta y
\end{aligned}
$$

$$
\begin{aligned}
\underline{(\lambda x.y)\,((\lambda z.zz)\,(\lambda w.w))} &\to_\beta y[(\lambda z.zz)\,(\lambda w.w)/x] \\
&\equiv y
\end{aligned}
$$

Observăm că:

- reducerea unui redex poate crea noi redex-uri
- reducerea unui redex poate șterge alte redex-uri
- numărul de pași necesari până a atinge o formă normală poate varia, în funcție de ordinea în care sunt reduse redex-urile
- rezultatul final pare că nu a depins de alegerea redex-urilor

β-formă normală

Totuși, există lambda termeni care nu pot fi reduși la o β-formă normală (evaluarea nu se termină).

$$\begin{aligned} \underline{(\lambda x.x\,x)\,(\lambda x.x\,x)} \quad &\to_\beta \quad (\lambda x.x\,x)\,(\lambda x.x\,x) \\ &\to_\beta \quad \dots \end{aligned}$$

Observați că lungimea unui termen nu trebuie să scadă în procesul de β-reducție; poate crește sau rămâne neschimbat.

β-formă normală

Există lambda termeni care deși pot fi reduși la o formă normală, pot să nu o atingă niciodată.

$$
\begin{aligned}
\underline{(\lambda xy.y)\,((\lambda x.x\,x)\,(\lambda x.x\,x))}\,(\lambda z.z) &\to_\beta \underline{(\lambda y.y)\,(\lambda x.x)} \\
&\to_\beta \lambda x.x
\end{aligned}
$$

$$
(\lambda xy.y)\,(\underline{(\lambda x.x\,x)\,(\lambda x.x\,x)})\,(\lambda z.z) \to_\beta (\lambda xy.y)\,((\lambda x.x\,x)\,(\lambda x.x\,x))\,(\lambda z.z)
$$

Contează strategia de evaluare.

β-formă normală

Notăm cu $M \twoheadrightarrow_\beta M'$ faptul că M poate fi β-redus până la M' în 0 sau mai mulți pași (închiderea reflexivă și tranzitivă a relației $\rightarrow_\beta$).

M este slab normalizabil *(weakly normalising)* dacă există N în formă normală astfel încât $M \twoheadrightarrow_\beta N$.

M este puternic normalizabil *(strong normalising)* dacă nu există reduceri infinite care încep din M.

Orice termen puternic normalizabil este și slab normalizabil.

Example

$(\lambda x.y)\,((\lambda z.zz)\,(\lambda w.w))$ este puternic normalizabil.

$(\lambda xy.y)\,((\lambda x.x\,x)\,(\lambda x.x\,x))\,(\lambda z.z)$ este slab normalizabil, dar nu puternic normalizabil.

Confluența β-reducției

Teorema Church-Rosser. Dacă $M \twoheadrightarrow_\beta M_1$ și $M \twoheadrightarrow_\beta M_2$ atunci există M' astfel încât $M_1 \twoheadrightarrow_\beta M'$ și $M_2 \twoheadrightarrow_\beta M'$.

$$\begin{tikzcd} & M \arrow[dl, two heads] \arrow[dr, two heads] & \\ M_1 \arrow[dr, two heads, dashed] & & M_2 \arrow[dl, two heads, dashed] \\ & M' & \end{tikzcd}$$

Consecință. Un lambda termen are cel mult o β-formă normală (modulo α-echivalență).

Exercițiu. Verificați dacă termenii de mai jos pot fi aduși la o β-formă normală:

1. $(\lambda x.x)\, M$
2. $(\lambda xy.x)\, M\, N$
3. $(\lambda x.x\, x)\, (\lambda y.y\, y\, y)$

Exercițiu. Verificați dacă termenii de mai jos pot fi aduși la o β-formă normală:

1. $(\lambda x.x)\, M$ — Corect: M
2. $(\lambda xy.x)\, M\, N$ — Corect: M
3. $(\lambda x.x\ x)\, (\lambda y.y\ y\ y)$ — Corect: $(\lambda y.y\ y\ y)\, (\lambda y.y\ y\ y)\, (\lambda y.y\ y\ y) \ldots$

Strategii de evaluare

Strategii de evaluare

De cele mai multe ori, există mai mulți pași de β-reducție care pot fi aplicați unui termen. Cum alegem ordinea? Contează ordinea?

O strategie de evaluare ne spune în ce ordine să facem pașii de reducție.

Lambda calculul nu specifică o strategie de evaluare, fiind nedeterminist. O strategie de evaluare este necesară în limbaje de programare reale pentru a rezolva nedeterminismul.

Strategia normală (normal order)

Strategia normală = *leftmost-outermost*

(alegem redex-ul cel mai din stânga și apoi cel mai din exterior)

- dacă M_1 și M_2 sunt redex-uri și M_1 este un subtermen al lui M_2, atunci M_1 nu va fi următorul redex ales
- printre redex-urile care nu sunt subtermeni ai altor redex-uri (și sunt incomparabili față de relația de subtermen), îl alegem pe cel mai din stânga.

Dacă un termen are o formă normală, atunci strategia normală va converge la ea.

$$\begin{aligned} \underline{(\lambda xy.y)\,((\lambda x.x\,x)\,(\lambda x.x\,x))}\,(\lambda z.z) &\to_\beta \underline{(\lambda y.y)\,(\lambda x.x)} \\ &\to_\beta \lambda x.x \end{aligned}$$

Strategia aplicativă (applicative order)

Strategia aplicativă = *leftmost-innermost*
(alegem redex-ul cel mai din stânga și apoi cel mai din interior)

- dacă M_1 și M_2 sunt redex-uri și M_1 este un subtermen al lui M_2, atunci M_2 nu va fi următorul redex ales
- printre redex-urile care nu sunt subtermeni ai altor redex-uri (și sunt incomparabili față de relația de subtermen), îl alegem pe cel mai din stânga.

$$(\lambda xy.y)\,\underline{((\lambda x.x\,x)\,(\lambda x.x\,x))}\,(\lambda z.z) \quad \to_\beta \quad (\lambda xy.y)\,((\lambda x.x\,x)\,(\lambda x.x\,x))\,(\lambda z.z)$$

Strategii în programare funcțională

În limbaje de programare funcțională, în general, reducerile din corpul unei λ-abstractizări nu sunt efectuate (deși anumite compilatoare optimizate pot face astfel de reduceri în unele cazuri).

Strategia *call-by-name* (CBN) = strategia normală fără a face reduceri în corpul unei λ-abstractizări

Strategia *call-by-value* (CBV) = strategia aplicativă fără a face reduceri în corpul unei λ-abstractizări

Majoritatea limbajelor de programare funcțională folosesc CBV, excepție făcând Haskell.

CBN vs CBV

O valoare este un λ-term pentru care nu există β-reducții date de strategia de evaluare considerată.

De exemplu, $\lambda x.x$ este mereu o valoare, dar $(\lambda x.x)\ 1$ nu este.

Sub CBV, funcțille pot fi apelate doar prin valori (argumentele trebuie să fie complet evaluate). Astfel, putem face β-reducția $(\lambda x.M)\ N \twoheadrightarrow_\beta M[N/x]$ doar dacă N este valoare.

Sub CBN, amânăm evaluarea argumentelor cât mai mult posibil, făcând reducții de la stânga la dreapta în expresie. Aceasta este strategia folosită în Haskell.

CBN este o formă de evaluare leneșă *(lazy evaluation)*: argumentele funcțiilor sunt evaluate doar când sunt necesare.

CBN vs CBV

Example

Considerăm 3 și *succ* primitive.

Strategia CBV:

$$
\begin{aligned}
(\lambda x.succ\, x)\,((\lambda y.succ\, y)\,3) &\twoheadrightarrow_\beta (\lambda x.succ\, x)\,(succ\,3)\\
&\to (\lambda x.succ\, x)\,4\\
&\twoheadrightarrow_\beta succ\,4\\
&\to 5
\end{aligned}
$$

Strategia CBN:

$$
\begin{aligned}
(\lambda x.succ\, x)\,((\lambda y.succ\, y)\,3) &\twoheadrightarrow_\beta succ\,((\lambda y.succ\, y)\,3)\\
&\twoheadrightarrow_\beta succ\,(succ\,3)\\
&\to succ\,4\\
&\to 5
\end{aligned}
$$

Quiz time!

https://tinyurl.com/C03-Quiz1

Pe săptămâna viitoare!