Strategii de evaluare

Reamintim că lambda-termenii sunt arbori generați de următoarea gramatică:

$$\begin{array}{rll} t ::= & x & x \text{ este un identificator} \\ \mid & t\ t & \text{lambda aplicație} \\ \mid & \lambda x.t & \text{lambda abstracție.} \end{array}$$

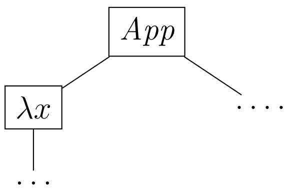
În lambda-calcul, există o singură regulă care modelează un pas de calcul, regulă care se numește beta-reducere.

Regula de β-reducere este următoarea:

$$(\lambda x.t)\ t' \to_\beta t[\![x/t']\!].$$

Cu alte cuvinte, singura regulă de calcul este aplicarea unei funcții $((\lambda x.t))$ pe un argument (t'). Observați folosirea substituției care evită capturarea.

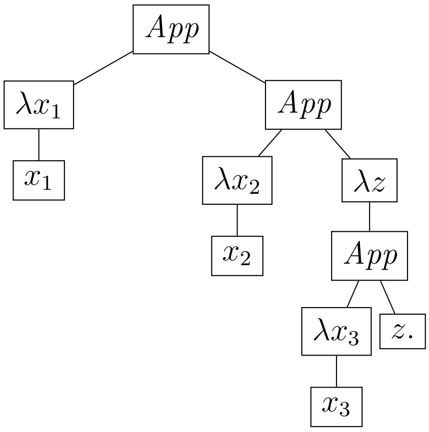
Lambda-termenii de forma $(\lambda x.t)\ t'$ se numesc *redex*-uri (de la engl. *reducible expression*). Sub formă de arbore, un redex arată în felul următor:



Un lambda-termen poate conține unul sau mai multe redex-uri. Spre exemplu, lambda-termenul următor conține trei redexuri:

$$(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big).$$

Arborele de sintaxă abstractă asociat lambda-termenului de mai sus este:



În termenul de mai sus, toți cei trei subtermeni cu rădăcina *App* sunt redex-uri.

Q: Într-un calcul, cum alegem redex-uri pe care îl vom reduce la următorul pas?

A: Depinde de strategia de evaluare pe care o folosesc.

1 Strategia *Full Beta-Reduction*

Această strategie este cea mai generală, și permite reducerea *oricărui* redex.

Folosind această strategie, într-un pas de evaluare, termenul

$$(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big)$$

se poate reduce la oricare dintre următorii trei termeni (în funcție de redex-ul care este redus):

1. $\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big)$;
2. $(\lambda x_1.x_1)\ (\lambda z.(\lambda x_3.x_3)\ z)$;
3. $(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.z)\Big)$.

În fiecare dintre cele trei cazuri, există câte două redex-uri care se pot aplica mai departe, după cum urmează:

1. $\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big)$ se reduce într-un pas la:
   (a) $(\lambda z.(\lambda x_3.x_3)\ z)$ sau la
   (b) $\Big((\lambda x_2.x_2)\ (\lambda z.z)\Big)$;

;

2. $(\lambda x_1.x_1)\ (\lambda z.(\lambda x_3.x_3)\ z)$ se reduce într-un pas la:

   (a) $(\lambda z.(\lambda x_3.x_3)\ z)$ sau la

   (b) $(\lambda x_1.x_1)\ (\lambda z.z)$;

   ;

3. $(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.\ z)\Big)$ se reduce într-un pas la:

   (a) $\Big((\lambda x_2.x_2)\ (\lambda z.z)\Big)$ sau la

   (b) $(\lambda x_1.x_1)\ (\lambda z.z)$.

   .

Într-un final, fiecare dintre cei 6 lambda-termeni obținuți în doi pași de reducere conține câte un singur redex, iar toți cei 6 lambda-termeni se reduc într-un singur pas la $\lambda z.z$, care este o *formă normală* (lambda-termen în care nu mai există nicio reducere de aplicat).

Cu alte cuvinte, cu strategia *full beta-reduction*, oricare dintre următoarele 6 calcule este valid:

1. Calculul I:
$$\begin{array}{l}(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\ (\lambda z.(\lambda x_3.x_3)\ z) \to\\ \lambda z.z \not\to;\end{array}$$

2. Calculul II:
$$\begin{array}{l}(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\ \Big((\lambda x_2.x_2)\ (\lambda z.z)\Big) \to\\ \lambda z.z \not\to;\end{array}$$

3. Calculul III:
$$\begin{array}{l}(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\ (\lambda x_1.x_1)\ (\lambda z.(\lambda x_3.x_3)\ z) \to\\ (\lambda z.(\lambda x_3.x_3)\ z) \to\\ \lambda z.z \not\to;\end{array}$$

4. Calculul IV:

$$\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\
&(\lambda x_1.x_1)\ (\lambda z.(\lambda x_3.x_3)\ z) \to\\
&(\lambda x_1.x_1)\ (\lambda z.z) \to\\
&\lambda z.z \not\to;
\end{aligned}$$

5. Calculul V:

$$\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.z)\Big) \to\\
&(\lambda x_1.x_1)\ (\lambda z.z) \to\\
&\lambda z.z \not\to;
\end{aligned}$$

6. Calculul VI:

$$\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.z)\Big) \to\\
&(\lambda x_1.x_1)\ (\lambda z.z) \to\\
&\lambda z.z \not\to .
\end{aligned}$$

În acest sens, această strategie este cea mai generală cu putință.

2 Strategia Normal Order

Această strategie (*Normal Order*) permite doar reducerea redex-ului cel mai din stânga și mai de sus.

Pentru exemplul nostru, singurul calcul permis de strategia *normal order* este calculul I de mai sus:

$$\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\
&\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to\\
&(\lambda z.(\lambda x_3.x_3)\ z) \to\\
&\lambda z.z \not\to .
\end{aligned}$$

Această strategie de evaluare definește o funcție parțială, în sensul în care dacă termenul nu este în formă normală, acesta are *exact* un succesor (până la alfa-echivalență). Cu alte cuvinte, această strategie permite exact un calcul pentru un lambda-termen fixat.

3 Strategia Applicative Order

Această strategie (*Applicative Order*) permite doar reducerea redex-ului cel mai din stânga. Dacă sunt mai multe redexuri la fel de în stânga, este ales redexul cel mai de jos. Este duală strategiei *normal order*.

4 Strategia *Call by Name*

Strategia *Call by Name* (CBN) este similară cu strategia *Normal Order*, dar nu sunt permise reducerile în interiorul unei lambda-abstracții.

Pentru exemplul de mai sus, calculul permis de această strategie este:

$$\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to \\
&\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to \\
&(\lambda z.(\lambda x_3.x_3)\ z) \not\to .
\end{aligned}$$

Observați că termenul $(\lambda z.(\lambda x_3.x_3)\ z)$ este în formă normală relativ la strategia CBN, deși nu este în formă normală în general.

Observați că primul pas este apelul funcției $(\lambda x_1.x_1)$ pentru argumentul $\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big)$. Observați de asemenea că argumentul ar putea fi redus (deoarece conține redexuri), dar strategia CBN forțează aplicarea funcției $(\lambda x_1.x_1)$ fără a reduce (calcula) mai întâi argumentul. Din acest motiv, această strategie de evaluare este o strategie *nestrictă* (strategiile nestricte sunt strategiile în care argumentele nu sunt complet evaluate înainte de apelul unei funcții).

Un alt exemplu de reducere CBN:

$$\begin{aligned}
&(\lambda x_1.\lambda x_2.x_2)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&\lambda x_2.x_2 \not\to .
\end{aligned}$$

Observați potențialul avantaj al unui calcul prin strategia call-by-name: parametrul formal x_1 este înlocuit direct cu argumentul $\Big((\lambda x.x)\ (\lambda y.y)\Big)$, dar fiindcă x_1 nu este folosit mai departe în calcul, nu mai sunt necesari pașii de calcul pentru argumentul $\Big((\lambda x.x)\ (\lambda y.y)\Big)$ în sine.

Totuși, există și un potențial dezavantaj ilustrat de următorul calcul:

$$
\begin{aligned}
&(\lambda x_1.x_1\ x_1)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&\Big((\lambda x.x)\ (\lambda y.y)\Big)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&(\lambda y.y)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&\Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&\lambda y.y \not\to .
\end{aligned}
$$

Fiindcă argumentul x_1 este folosit de două ori în corpul funcției $\lambda x_1.x_1\ x_1$, argumentul efectiv $\Big((\lambda x.x)\ (\lambda y.y)\Big)$ va ajunge să fie evaluat de două ori (o dată de la rândul 2 la 3, altă dată de la rândul 4 la 5).

Haskell folosește o strategie echivalentă cu call-by-name, strategie numită *call-by-need*, care evită dezavantajul de mai sus (fiecare termen este evaluat cel mult o dată).

5 Strategia *Call by Value*

Cele mai multe limbaje de programare preferă o strategie numită *Call by Value*, în care argumentele unei funcții sunt evaluate complet înainte să fie evaluat corpul funcției.

Pentru a defini strategia call-by-value, avem nevoie să definim mai întâi ce este o *valoare* (*value* în engleză).

În general (nu neapărat în lambda-calcul), o valoare este o expresie care nu mai poate fi redusă. De exemplu, expresiile 7 și *True* sunt valori, în timp ce expresiile $2 + (3 + 2)$ și $(3 < 4) \wedge (4 < 5)$ nu sunt valori (deoarece se mai pot face pași de calcul).

În lambda-calcul nu există predefinite numere sau booleeni, așadar singurele valori sunt lambda-abstracțiile. De exemplu, $\lambda x.x$ și $\lambda x.((\lambda y.y)x)$ sunt valori, în timp ce $(\lambda x.x)\ (\lambda y.y)$ nu este o valoare.

În strategie call-by-value, singurul redex permis a fi redus este redex-ul cel mai de sus a cărui parte dreapta este o valoare (la fel ca la call-by-name, nu sunt permise reduceri *sub* o lambda-abstracție).

Pentru exemplul nostru inițial, singurul calcul permis de strategia call-by-value este:

$$
\begin{aligned}
&(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big) \to \\
&(\lambda x_1.x_1)\ (\lambda z.(\lambda x_3.x_3)\ z) \to \\
&(\lambda z.(\lambda x_3.x_3)\ z) \not\to .
\end{aligned}
$$

Observați că, spre deosebire de call-by-name, în strategia call-by-value este evaluat întâi argumentul $\Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big)$ al funcției $(\lambda x_1.x_1)$, înainte de a evalua corpul funcției.

Acest lucru poate fi un dezavantaj (față de call-by-name) din punct de vedere al numărului de pași de calcul, după cum este ilustrat în următorul exemplu:

$$\begin{aligned}
&(\lambda x_1.\lambda x_2.x_2)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&(\lambda x_1.\lambda x_2.x_2)\ (\lambda y.y) \to \\
&\lambda x_2.x_2 \not\to .
\end{aligned}$$

Dar poate fi și un avantaj:

$$\begin{aligned}
&(\lambda x_1.x_1\ x_1)\ \Big((\lambda x.x)\ (\lambda y.y)\Big) \to \\
&(\lambda x_1.x_1\ x_1)\ (\lambda y.y) \to \\
&(\lambda y.y)\ (\lambda y.y) \to \\
&\lambda y.y \not\to .
\end{aligned}$$

Această strategie este o strategie *strictă*, deoarece argumentele funcțiilor sunt evaluate complet înainte de a se evalua funcția în sine.