Latihan - Simply Typed λ Calculus

Disarikan oleh Ade Azurat dari berbagai sumber.

2019-11-20

1 Church Numeral

- $0 \equiv (\lambda sz.z)$
- $1 \equiv (\lambda sz.s(z))$
- $2 \equiv (\lambda sz.s(s(z)))$
- $3 \equiv (\lambda sz.s(s(s(z))))$
- dan seterusnya
- $S \equiv (\lambda wyx.y(wyx))$
- Fungsi successor S dapat memodelkan bilangan secara unary: 1 = S(0), 2=S(S(0)) dan seterusnya.
- Misalkan kita ingin menjumlahkan

$$2 + 3$$

- dalam $\lambda Calculus$ dapat dimodelkan dengan

$$2S3$$

- $2S3 \equiv (\lambda sz.s(s(z)))(\lambda wyx.y(wyx))(\lambda uv.u(u(u(v))))$
- Perkalian adalah pengulangan penjumlahan dan dapat dimodelkan dengan

$$\lambda xyz.x(yz)$$

- Perkalian

$$0x1$$

  dimodelkan dengan

$$(\lambda xyz.x(yz))01$$

- Pada penulisan kali ini kita perkenalkan 'literal 0 dan 1' yang kelak akan dikembalikan kepada definisi representasinya di $\lambda Calculus$

- $0 * 1$
  $\equiv (\lambda xyz.x(yz))01$ representasi dari $0 * 1$
  $\equiv (\lambda yz.0(yz))1$ aplikasi $0, x := 0$
  $\equiv (\lambda z.0(1z))$ aplikasi $1, y := 1$
  $\equiv (\lambda z.(\lambda s.(\lambda t.t))(1z))$ membuka definisi literal $0 \equiv (\lambda s.(\lambda t.t))$
  $\equiv (\lambda z.(\lambda s.(\lambda t.t))(1z))$ aplikasi $(1z), s := (1z)$
  $\equiv (\lambda z.(\lambda t.t)))$ definisi 0
  $\equiv 0$

2 Church Boolean

T (true) dinyatakan dengan $\lambda xy.x$

F (false) dinyatakan dengan $\lambda xy.y$

Conjuntion (and) ($\wedge$) dinyatakan dengan

$$\wedge \equiv \lambda xy.xy(\lambda uv.v) \equiv \lambda xy.xyF$$

Disjunction (or) ($\vee$) dinyatakan dengan

$$\vee \equiv \lambda xy.x(\lambda uv.u)y \equiv \lambda xy.xTy$$

Negation (not) ($\neg$) dinyatakan dengan

$$\lambda x.x(\lambda uv.v)(\lambda ab.a) \equiv \lambda x.xFT$$

Test zero:

$$Z \equiv \lambda x.xF\neg F$$

Representasi *church numeral*, menyatakan bilangan sebagai enumerasi yang merupakan fungsi menerima fungsi dan argumen kemudian menerapkan fungsi tersebut terus menerus sebanyak enumerasinya.

Ingat bahwa bila fungsi F diaplikasikan dengan apapun, maka akan menghasilkan fungsi identitas I.

$$Fa \equiv (\lambda xy.y)a \equiv \lambda y.y \equiv I$$

Perhatikan fungsi Z diaplikasi dengan bilangan bukan 0, misalkan N.

$ZN \equiv (\lambda x.xF\neg F)N$ definisi Z dan $\beta - reduction$
$\equiv NF\neg F$ karena F diaplikasi dengan $\neg F$ sebanyak N kali, hasilnya I
$\equiv IF$
$\equiv F$

3 Pair

Sebuah $pair(a, b)$ dinyatakan dalam $\lambda Calculus$ dengan

$$(a, b) \equiv \lambda z.zab$$

Untuk dapat menggenerate pair ke-n, didefinisikan fungsi generator:

$$\Phi \equiv (\lambda pz.z(S(pT))(pT))$$

Predecessor (Nilai sebelumnya (n-1))

$$P \equiv (\lambda n.n\ \Phi(\lambda z.z00)F)$$

Dengan memanfaatkan definisi predecessor P kita bisa definisikan "predikat lebih besar sama dengan" ($\geq$) kita simbolkan dengan G sebagai berikut:

$$G \equiv (\lambda xy.Z(xPy))$$

Untuk mendefinisikan persamaan ($equality$) dari bilangan didefinisikan E:

$$E \equiv (\lambda xy. \wedge (Z(xPy))(Z(yPx)))$$

yang menyatakan bahwa bila $x \geq y$ dan $y \geq x$ maka $x = y$

4 Rekursif dan Y Combinator

Definisi rekursif dimodelkan dalam $\lambda Calculus$ dengan kombinator yang disebut $Y - Compinator$. Kombinator tersebut me-regenerate sebuah fungsi rekursif menjadi fungsi baru namun setara dan bisa di-evaluasi secara finite (Kaitan nya dengan Fixed Point, silahkan baca literatur lanjutan). Kombinator Y, didefinisikan sebagai berikut:

$$Y \equiv (\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))$$

Bagaimana penerapan-nya? Misalkan ada sebuah fungsi rekursif R.

$$\begin{array}{rll} YR \equiv & ((\lambda x.R(xx))(\lambda x.R(xx))) & definisi \\ \equiv & R((\lambda x.R(xx))(\lambda x.R(xx))) & \beta - reduction \\ \equiv & R(YR) & rekursif \end{array}$$

Dengan kata lain, bila fungsi Y diaplikasikan dengan sebuah fungsi R, maka akan menghasilkan pemanggilan rekursi dari fungsi R tersebut.

Contoh:

$$\Sigma_{i=0}^{n} i = n + \Sigma_{i=0}^{n-1} i$$

dalam pseudo code bahasa pemograman biasa ditulis secara rekursif sebagai berikut:

sigma (0,n,i) = n + sigma(0, n-1, i);

Fungsi rekursif tersebut diberi nama R dan dimodelkan dalam $\lambda Calculus$ menjadi:

$$R \equiv (\lambda rn.Zn0(nS(r(Pn))))$$

Misalkan dengan parameter $n = 3$, dan definisi dari Y, penerapan rekursifnya menjadi:

$$YR3 = R(YR)3 = Z30(3S(YR(P3)))$$

Karena 3 tidak sama dengan 0, maka menjadi:

$$3S(YR2)$$

Bila kembali diterapkan definisi Y, maka akan didapat:

$$3S(YR2) \equiv 3S(2S(YR1)) \equiv 3S(2S(1S0)) \equiv 6$$

$$\frac{}{\Gamma, x : T \vdash x : T} \quad (1)$$

$$\frac{\Gamma \vdash t : U \to T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T} \quad (2)$$

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash (\lambda x : U \cdot t) : U \to T} \quad (3)$$

$$\dfrac{\dfrac{\dfrac{}{avg : \cdots, f : \beta \vdash f : \delta \to \alpha} \qquad \dfrac{}{avg : \cdots, f : \cdots \vdash 3 : \delta}}{avg : \cdots, f : \beta \vdash f\,3 : \alpha}}{avg : \cdots \vdash (\lambda f : \beta \cdot f\,3) : \beta \to \alpha} \qquad \dfrac{\dfrac{}{avg : \mathbf{Num} \to \mathbf{Num} \to \mathbf{Num} \vdash avg : \gamma \to \beta} \qquad \dfrac{}{avg : \cdots \vdash 2 : \gamma}}{avg : \mathbf{Num} \to \mathbf{Num} \to \mathbf{Num} \vdash avg\,2 : \beta}}{avg : \mathbf{Num} \to \mathbf{Num} \to \mathbf{Num} \vdash (\lambda f : \beta \cdot f\,3)(avg\,2) : \alpha}$$

Analisa serupa dengan langkah-langkah tersebut diatas dapat digunakan juga untuk menguji apakah sebuah fungsi rekursif akan berhenti atau tidak.

5 Latihan

1. Model kan dan evaluasi dalam $\lambda Calculus$ perkalian $1 * 2$?
2. Model kan dan evaluasi perkalian dengan Σ. $m * n = \Sigma_{i=0}^{m} n$
3. Uji apakah: $\lambda(x : Num).x3$ memiliki tipe: Num $\to$ Num.
4. Tentukan tipe dari: $\lambda x.x3$.