看看程式語言學在幹嘛

柯向上 Josh Ko

中央研究院資訊科學研究所

FLOLAC 識別設計溝通會議，2020 年 3 月 10 日

程式語言學

以符號（語言）精準簡練地表達計算規則（程式）

厲害：嚴格證明各種「好」性質

易用：作為好工具，協助人思考與解決問題

內涵：常與數學、邏輯學有密切對應

厲害

規則、符號、好性質

拍賣網站購物

「先詢問有無存貨，賣家回覆有貨才可下標。下標後將收件資訊告知賣家，完成匯款後賣家出貨。」

```
買家動作：    ![請問還有沒有貨]
              ▹{ 缺貨 : stop
                 有貨 : if 老婆說可以買
                        then ◃ 下標
                             ![收件資訊]
                             ![錢]
                             ?(貨)
                             stop
                        else ◃ 算了
                             stop  }
```

! 送
? 收
▹ 對方選擇
◃ 自己選擇

拍賣網站購物

「先詢問有無存貨，賣家回覆有貨才可下標。下標後將收件資訊告知賣家，完成匯款後賣家出貨。」

賣家動作：

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
     ▹{ 算了 : stop
        下標 : ?(收件資訊)
               ?(錢)
               ![貨]
               stop  }
else ◃ 缺貨
     stop
```

! 送

? 收

▹ 對方選擇

◃ 自己選擇

程式的合法長相（語言）

$$
\begin{array}{lcll}
P & ::= & \texttt{request}\ a(k)\ \texttt{in}\ P & \text{session request} \\
& | & \texttt{accept}\ a(k)\ \texttt{in}\ P & \text{session acceptance} \\
& | & k![\tilde{e}]; P & \text{data sending} \\
& | & k?(\tilde{x})\ \texttt{in}\ P & \text{data reception} \\
& | & k \lhd l; P & \text{label selection} \\
& | & k \rhd \{l_1 : P_1 \| \cdots \| l_n : P_n\} & \text{label branching} \\
& | & \texttt{throw}\ k[k']; P & \text{channel sending} \\
& | & \texttt{catch}\ k(k')\ \texttt{in}\ P & \text{channel reception} \\
& | & \texttt{if}\ e\ \texttt{then}\ P\ \texttt{else}\ Q & \text{conditional branch} \\
& | & P \mid Q & \text{parallel composition} \\
& | & \texttt{inact} & \text{inaction} \\
& | & (\nu u)P & \text{name/channel hiding} \\
& | & \texttt{def}\ D\ \texttt{in}\ P & \text{recursion} \\
& | & X[\tilde{e}\tilde{k}] & \text{process variables} \\
D & ::= & X_1(\tilde{x}_1\tilde{k}_1) = P_1\ \texttt{and} \cdots \texttt{and}\ X_n(\tilde{x}_n\tilde{k}_n) = P_n & \text{declaration for recursion}
\end{array}
$$

Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo [1998]. Language primitives and type discipline for structured communication-based programming. In *European Symposium on Programming*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer. https://doi.org/10.1007/BFb0053567.

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
               ![收件資訊]
               ![錢]
               ?(貨)
               stop
          else ◃ 算了
               stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
     ▹{ 算了 : stop
        下標 : ?(收件資訊)
               ?(錢)
               ![貨]
               stop  }
else ◃ 缺貨
     stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
               ![收件資訊]
               ![錢]
               ?(貨)
               stop
          else ◃ 算了
               stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
     ▹{ 算了 : stop
        下標 : ?(收件資訊)
               ?(錢)
               ![貨]
               stop  }
else ◃ 缺貨
     stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
               ![收件資訊]
               ![錢]
               ?(貨)
               stop
          else ◃ 算了
               stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
     ▹{ 算了 : stop
        下標 : ?(收件資訊)
               ?(錢)
               ![貨]
               stop  }
else ◃ 缺貨
     stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                 ![收件資訊]
                 ![錢]
                 ?(貨)
                 stop
          else ◃ 算了
                 stop   }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop   }
else ◃ 缺貨
       stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                 ![收件資訊]
                 ![錢]
                 ?(貨)
                 stop
          else ◃ 算了
                 stop   }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop   }
else ◃ 缺貨
       stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                ![收件資訊]
                ![錢]
                ?(貨)
                stop
          else ◃ 算了
                stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop  }
else ◃ 缺貨
       stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
               then ◃ 下標
                      ![收件資訊]
                      ![錢]
                      ?(貨)
                      stop
               else ◃ 算了
                      stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
          ▹{ 算了 : stop
              下標 : ?(收件資訊)
                     ?(錢)
                     ![貨]
                     stop  }
else ◃ 缺貨
          stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                 ![收件資訊]
                 ![錢]
                 ?(貨)
                 stop
          else ◃ 算了
                 stop   }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop   }
else ◃ 缺貨
       stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                 ![收件資訊]
                 ![錢]
                 ?(貨)
                 stop
          else ◃ 算了
                 stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop  }
else ◃ 缺貨
       stop
```

賣家

實際進行購物

```
![請問還有沒有貨]
▷{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◁ 下標
               ![收件資訊]
               ![錢]
               ?(貨)
               stop
          else ◁ 算了
               stop  }
```

買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◁ 有貨
     ▷{ 算了 : stop
        下標 : ?(收件資訊)
               ?(錢)
               ![貨]
               stop  }
else ◁ 缺貨
     stop
```

賣家

定義程式如何執行

$$
\begin{array}{ll}
[\textsc{Link}] & (\texttt{accept}\ a(k)\ \texttt{in}\ P_1) \mid (\texttt{request}\ a(k)\ \texttt{in}\ P_2)\ \rightarrow\ (\nu k)(P_1 \mid P_2) \\
[\textsc{Com}] & (k![\tilde{e}];P_1) \mid\ (k?(\tilde{x})\ \texttt{in}\ P_2)\ \rightarrow\ P_1 \mid P_2[\tilde{c}/\tilde{x}] \qquad (\tilde{e} \downarrow \tilde{c}) \\
[\textsc{Label}] & (k \lhd l_i;P) \mid (k \rhd \{l_1 : P_1 \| \cdots \| l_n : P_n\})\ \rightarrow\ P \mid\ P_i \quad (1 \leq i \leq n) \\
[\textsc{Pass}] & (\texttt{throw}\ k[k'];P_1) \mid\ (\texttt{catch}\ k(k')\ \texttt{in}\ P_2)\ \rightarrow\ P_1 \mid\ P_2 \\
[\textsc{If1}] & \texttt{if}\ e\ \texttt{then}\ P_1\ \texttt{else}\ P_2\ \rightarrow\ P_1 \qquad (e \downarrow \texttt{true}) \\
[\textsc{If2}] & \texttt{if}\ e\ \texttt{then}\ P_1\ \texttt{else}\ P_2\ \rightarrow\ P_2 \qquad (e \downarrow \texttt{false}) \\
[\textsc{Def}] & \texttt{def}\ D\ \texttt{in}\ (X[\tilde{e}\tilde{k}] \mid Q)\ \rightarrow\ \texttt{def}\ D\ \texttt{in}\ (P[\tilde{c}/\tilde{x}] \mid Q)\ (\tilde{e} \downarrow \tilde{c}, X(\tilde{x}\tilde{k}) = P \in D) \\
[\textsc{Scop}] & P \rightarrow P' \ \Rightarrow\ (\nu u)P \rightarrow (\nu u)P' \\
[\textsc{Par}] & P \rightarrow P' \ \Rightarrow\ P \mid Q\ \rightarrow\ P' \mid Q \\
[\textsc{Str}] & P \equiv P' \text{ and } P' \rightarrow Q' \text{ and } Q' \equiv Q\ \Rightarrow\ P \rightarrow Q
\end{array}
$$

Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo [1998]. Language primitives and type discipline for structured communication-based programming. In *European Symposium on Programming*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer. https://doi.org/10.1007/BFb0053567.

配對錯誤

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標
                 ![收件資訊]
                 ?(貨)
                 ![錢]
                 stop
          else ◃ 算了
                 stop  }
```

以為可以貨到付款的買家

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨
       ▹{ 算了 : stop
          下標 : ?(收件資訊)
                 ?(錢)
                 ![貨]
                 stop  }
else ◃ 缺貨
       stop
```

賣家

通訊協議 / 訊程型別

「先詢問有無存貨，賣家回覆有貨才可下標。下標後將收件資訊告知賣家，完成匯款後賣家出貨。」

買家通訊規則：

```
!詢問存貨
&{ 缺貨 ： end
   有貨 ： +{ 下標 ： !收件資訊 !錢 ?貨 end
              算了 ： end } }
```

! 送
? 收
& 對方所有選項
+ 自己所有選項

保證沒有配對錯誤

買家動作

```
![請問還有沒有貨]
▹{ 缺貨 : stop
   有貨 : if 老婆說可以買
          then ◃ 下標; ![收件資訊] ![錢] ?(貨) stop
          else ◃ 算了; stop  }
```

遵守

買家通訊規則

```
!詢問存貨
&{ 缺貨 : end
   有貨 : +{ 下標 : !收件資訊 !錢 ?貨 end
             算了 : end } }
```

保證沒有配對錯誤

!詢問存貨 買家通訊規則

&{ 缺貨 : end
　 有貨 : +{ 下標 : !收件資訊 !錢 ?貨 end
　　　　　　 算了 : end } }

對偶

?詢問存貨 賣家通訊規則

+{ 缺貨 : end
　 有貨 : &{ 下標 : ?收件資訊 ?錢 !貨 end
　　　　　　 算了 : end } }

保證沒有配對錯誤

賣家通訊規則

```
?詢問存貨
+{ 缺貨 : end
   有貨 : &{ 下標 : ?收件資訊 ?錢 !貨 end
            算了 : end } }
```

遵守

賣家動作

```
?(請問還有沒有貨)
if 查了一下有貨
then ◃ 有貨; ▹{ 算了 : stop
               下標 : ?(收件資訊) ?(錢) ![貨] stop }
else ◃ 缺貨; stop
```

型別安全定理

若兩個通訊程式遵守對偶的訊程型別，
則它們一起執行時不可能發生配對錯誤。

$$S \;::=\; \mathbf{nat} \mid \mathbf{bool} \mid \langle \alpha, \overline{\alpha} \rangle \mid s \mid \mu s.S$$

$$\begin{aligned}\alpha \;::=\; & \downarrow[\tilde{S}];\alpha \mid \downarrow[\alpha];\beta \mid \&\{l_1\colon \alpha_1,\ldots,l_n\colon \alpha_n\} \mid \mathbf{1} \mid \bot \\ & \mid \uparrow[\tilde{S}];\alpha \mid \uparrow[\alpha];\beta \mid \oplus\{l_1\colon \alpha_1,\ldots,l_n\colon \alpha_n\} \mid t \mid \mu t.\alpha\end{aligned}$$

$$\overline{\uparrow[\tilde{S}];\alpha} = \downarrow[\tilde{S}];\overline{\alpha} \quad \overline{\oplus\{l_i\colon \alpha_i\}} = \&\{l_i : \overline{\alpha_i}\} \quad \overline{\uparrow[\alpha];\beta} = \downarrow[\alpha];\overline{\beta} \quad \overline{\mathbf{1}} = \mathbf{1}$$

$$\overline{\downarrow[\tilde{S}];\alpha} = \uparrow[\tilde{S}];\overline{\alpha} \quad \overline{\&\{l_i\colon \alpha_i\}} = \oplus\{l_i : \overline{\alpha_i}\} \quad \overline{\downarrow[\alpha];\beta} = \uparrow[\alpha];\overline{\beta} \quad \overline{t} = t \quad \overline{\mu t.\alpha} = \mu t.\overline{\alpha}$$

We need the following notion: a k-*process* is a prefixed process with subject k (such as $k![\tilde{e}];P$ and $\texttt{catch}\ k(k')\ \texttt{in}\ P$). Next, a k-*redex* is a pair of dual k-processes composed by $|$, i.e. either of forms $(k![\tilde{e}];P \mid k?(x)\ \texttt{in}\ Q)$, $(k \lhd l;P \mid k \rhd \{l_1 : Q_1 \| \cdots \| l_n : Q_n\})$, or $(\texttt{throw}\ k[k'];P \mid \texttt{catch}\ k(k'')\ \texttt{in}\ Q)$. Then P is an *error* if $P \equiv \texttt{def}\ D\ \texttt{in}\ (\nu\tilde{u})(P'|R)$ where P' is, for some k, the $|$-composition of *either* two k-processes that do not form a k-redex, *or* three or more k-processes.

$$[\textsc{Acc}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \alpha}{\Theta;\Gamma, a : \langle \alpha, \overline{\alpha} \rangle \vdash \texttt{accept}\ a(k)\ \texttt{in}\ P \triangleright \Delta} \qquad [\textsc{Req}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \overline{\alpha}}{\Theta;\Gamma, a : \langle \alpha, \overline{\alpha} \rangle \vdash \texttt{request}\ a(k)\ \texttt{in}\ P \triangleright \Delta}$$

$$[\textsc{Send}]\ \frac{\Gamma \vdash \tilde{e} \triangleright \tilde{S} \quad \Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \alpha}{\Theta;\Gamma \vdash k![\tilde{e}];P \triangleright \Delta \cdot k :\uparrow[\tilde{S}];\alpha} \qquad [\textsc{Rcv}]\ \frac{\Theta;\Gamma \cdot \tilde{x} : \tilde{S} \vdash P \triangleright \Delta \cdot k : \alpha}{\Theta;\Gamma \vdash k?(\tilde{x})\ \texttt{in}\ P \triangleright \Delta \cdot k :\downarrow[\tilde{S}];\alpha}$$

$$[\textsc{Br}]\ \frac{\Theta;\Gamma \vdash P_1 \triangleright \Delta \cdot k : \alpha_1 \quad \cdots \quad \Theta;\Gamma \vdash P_n \triangleright \Delta \cdot k : \alpha_n}{\Theta;\Gamma \vdash\ k \rhd \{l_1 : P_1 \| \cdots \| l_n : P_n\} \triangleright \Delta \cdot k : \&\{l_1\colon \alpha_1,\ldots,l_n\colon \alpha_n\}}$$

$$[\textsc{Sel}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \alpha_j}{\Theta;\Gamma \vdash k \lhd l_j;P \triangleright \Delta \cdot k : \oplus\{l_1\colon \alpha_1,\ldots,l_n\colon \alpha_n\}} \quad (1 \le j \le n)$$

$$[\textsc{Thr}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \beta}{\Theta;\Gamma \vdash \texttt{throw}\ k[k'];P \triangleright \Delta \cdot k :\uparrow[\alpha];\beta \cdot k' : \alpha} \qquad [\textsc{Cat}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k : \beta \cdot k' : \alpha}{\Theta;\Gamma \vdash \texttt{catch}\ k(k')\ \texttt{in}\ P \triangleright \Delta \cdot k :\downarrow[\alpha];\beta}$$

$$[\textsc{Conc}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \quad \Theta;\Gamma \vdash Q \triangleright \Delta'}{\Theta;\Gamma \vdash P \mid Q \triangleright \Delta \circ \Delta'} (\Delta \asymp \Delta') \qquad [\textsc{If}]\ \frac{\Gamma \vdash e \triangleright \mathbf{bool} \quad \Theta;\Gamma \vdash P \triangleright \Delta \quad \Theta;\Gamma \vdash Q \triangleright \Delta}{\Theta;\Gamma \vdash \texttt{if}\ e\ \texttt{then}\ P\ \texttt{else}\ Q \triangleright \Delta}$$

$$[\textsc{NRes}]\ \frac{\Theta;\Gamma \cdot a : S \vdash P \triangleright \Delta}{\Theta;\Gamma \vdash (\nu a)P \triangleright \Delta} \qquad [\textsc{CRes}]\ \frac{\Theta;\Gamma \vdash P \triangleright \Delta \cdot k :\bot}{\Theta;\Gamma \vdash\ (\nu k)P \triangleright \Delta} \qquad [\textsc{Inact}]\ \Theta;\Gamma \vdash \texttt{inact} \triangleright \Delta$$

$$[\textsc{Var}]\ \frac{\Gamma \vdash \tilde{e} \triangleright \tilde{S}}{\Theta, X : \tilde{S}\tilde{\alpha};\Gamma \vdash X[\tilde{e}\tilde{k}] \triangleright \Delta \cdot \tilde{k} : \tilde{\alpha}} \qquad [\textsc{Def}]\ \frac{\Theta;\Gamma \cdot \tilde{x} : \tilde{S} \vdash P \triangleright \tilde{k} : \tilde{\alpha} \quad \Theta;\Gamma \vdash Q \triangleright \Delta}{\Theta\backslash X;\Gamma \vdash \texttt{def}\ X(\tilde{x}\tilde{k}) = P\ \texttt{in}\ Q \triangleright \Delta} (\Theta(X) = \tilde{S}\tilde{\alpha})$$

Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo [1998]. Language primitives and type discipline for structured communication-based programming. In *European Symposium on Programming*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer. https://doi.org/10.1007/BFb0053567.

易用

解決問題的好工具

解出一個數

問：小明買了 24 枝 9 元的鉛筆和 12 個 15 元的橡皮擦，請問小明總共要付多少錢？

答：$9 \times 24 + 15 \times 12 = 216 + 180 = 396$

```
  15 × 12
= (10 + 5) × (10 + 2)
= 100 + 20 + 50 + 10
= 180
```

```
   15
 × 12
 ----
   10
    2
    5
  1
 ----
  180
```

解出一個程式

寫下要做的事情

$$
\begin{array}{ll}
& \mathit{ordered} \cdot \mathit{perm} \\
\supseteq & \quad \{\text{since } \mathit{flatten} \text{ is a function}\} \\
& \mathit{ordered} \cdot \mathit{flatten} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \quad \{\text{claim: } \mathit{ordered} \cdot \mathit{flatten} = \mathit{flatten} \cdot \mathit{inordered} \text{ (see below)}\} \\
& \mathit{flatten} \cdot \mathit{inordered} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \quad \{\text{converses}\} \\
& \mathit{flatten} \cdot (\mathit{perm} \cdot \mathit{flatten} \cdot \mathit{inordered})^\circ \\
\supseteq & \quad \{\text{fusion, for an appropriate definition of } \mathit{split}\} \\
& \mathit{flatten} \cdot (\![\mathit{nil}, \mathit{split}^\circ]\!)^\circ .
\end{array}
$$

解出程式

符號計算

Richard Bird and Oege de Moor [1997]. *Algebra of Programming.* Prentice-Hall.

内涵

數學、邏輯學

函數

函數定義

$$f(x) = x^2 + 1$$

函數代入

$$f(5) \ = \ 5^2 + 1 \ = \ 26$$

無名函數

函數定義

$$\lambda x.\ x^2 + 1$$

函數代入

$$(\lambda x.\ x^2 + 1)\ 5 \ = \ 5^2 + 1 \ = \ 26$$

λ 算則

$$\Lambda ::= x \mid \lambda x.\ \Lambda \mid \Lambda\ \Lambda$$

$$(\lambda x.\ t)\ u \ = \ t[u/x]$$

2 + 3 = 5

$$
\begin{aligned}
& \overset{+}{(\lambda m.\ \lambda n.\ \lambda f.\ \lambda x.\ m\ f\ (n\ f\ x))}\ \overset{2}{(\lambda f.\ \lambda x.\ f\ (f\ x))}\ \overset{3}{(\lambda f.\ \lambda x.\ f\ (f\ (f\ x)))} \\
=\ & (\lambda n.\ \lambda f.\ \lambda x.\ (\lambda f.\ \lambda x.\ f\ (f\ x))\ f\ (n\ f\ x))\ (\lambda f.\ \lambda x.\ f\ (f\ (f\ x))) \\
=\ & \lambda f.\ \lambda x.\ (\lambda f.\ \lambda x.\ f\ (f\ x))\ f\ ((\lambda f.\ \lambda x.\ f\ (f\ (f\ x)))\ f\ x) \\
=\ & \lambda f.\ \lambda x.\ (\lambda f.\ \lambda x.\ f\ (f\ x))\ f\ ((\lambda x.\ f\ (f\ (f\ x)))\ x) \\
=\ & \lambda f.\ \lambda x.\ (\lambda f.\ \lambda x.\ f\ (f\ x))\ f\ (f\ (f\ (f\ x))) \\
=\ & \lambda f.\ \lambda x.\ (\lambda x.\ f\ (f\ x))\ (f\ (f\ (f\ x))) \\
=\ & \underset{5}{\lambda f.\ \lambda x.\ f\ (f\ (f\ (f\ (f\ x))))}
\end{aligned}
$$

型別

$$
\dfrac{
  \dfrac{
    \dfrac{}{f : \iota \to \iota, x : \iota \vdash f : \iota \to \iota}
    \qquad
    \dfrac{
      \dfrac{}{f : \iota \to \iota, x : \iota \vdash f : \iota \to \iota}
      \quad
      \dfrac{}{f : \iota \to \iota, x : \iota \vdash x : \iota}
    }{f : \iota \to \iota, x : \iota \vdash f\ x : \iota}
  }{
    \dfrac{f : \iota \to \iota, x : \iota \vdash f\ (f\ x) : \iota}{f : \iota \to \iota \vdash \lambda x.\ f\ (f\ x) : \iota \to \iota}
  }
}{\vdash \lambda f.\ \lambda x.\ f\ (f\ x) : (\iota \to \iota) \to \iota \to \iota}
$$

型別對應於邏輯

$$\frac{\Gamma \vdash t : \boxed{A \to B} \qquad \Gamma \vdash u : \boxed{A}}{\Gamma \vdash t\ u : \boxed{B}}$$

程式語言學

以符號（語言）精準簡練地表達計算規則（程式）

厲害：嚴格證明各種「好」性質

易用：作為好工具，協助人思考與解決問題

內涵：常與數學、邏輯學有密切對應