計算の理論

後半第3回「λ計算と型システム」

本日の内容

- λ計算の表現力（前回の復習）
  - データの表現
  - 不動点演算子と再帰
- λ計算の重要な性質
  - チャーチ・ロッサー性
  - 簡約戦略
- 型付きλ計算

ブール値と組の表現

- ブール値

  「true，false を受け取り、対応する要素を返す関数」として表現

  $T = \lambda t.\lambda f.t$　　$F = \lambda t.\lambda f.f$

  if e_1 then e_2 else e_3 = e_1 e_2 e_3 とすると...

  if T then e_2 else $e_3 \rightarrow^* e_2$

- 組

  makepair = $\lambda x.\lambda y.\lambda f.f\ x\ y$

  fst = $\lambda p.p(\lambda x.\lambda y.x)$: 組の1番目の要素の取り出し

  snd = $\lambda p.p(\lambda x.\lambda y.y)$: 組の2番目の要素の取り出し

  とすると...

  fst(makepair M N) $\rightarrow^*$ M

自然数の表現

- 「基本構成子(zeroとsuccessor)を受け取り、対応する値を返す関数」として表現

$$n = \lambda s.\lambda z.\ \underbrace{s\ (s\ \ldots\ (s}_{n}\ z)\ldots)$$

Plus = λm. λn. λs.λz. m s (n s z)

Mult= λm. λn. n (Plus m) 0

Exp = λm. λn. n (Mult m) 1

eqzero? = λn. n (λb.F) T

Pred = λn. snd(n (λ(x,y).(x+1,x)) (0,0))

Minus = λm. λn. n Pred m

本日の内容

- λ計算の表現力（前回の復習）
  - データの表現
  - 不動点演算子と再帰
- λ計算の重要な性質
  - チャーチ・ロッサー性
  - 簡約戦略
- 型付きλ計算

無限簡約列を持つ項

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

$\rightarrow [(\lambda x.\ x\ x)\ /\ x]\ (x\ x)$

$=\ \ (\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

$\rightarrow\ (\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

$\rightarrow\ (\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

$\rightarrow\ \ldots$

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$の変種

$(\lambda x.\ F\ (x\ x))\ (\lambda x.\ F(x\ x))$

$\rightarrow [(\lambda x.\ F\ (x\ x))\ /\ x]\ F(x\ x)$

$= F((\lambda x.\ F(x\ x))\ (\lambda x.\ F(x\ x)))$

$M_F = (\lambda x.\ F\ (x\ x))\ (\lambda x.\ F(x\ x))$とおくと…

$M_F \rightarrow F\ (M_F)$

$=_\beta$ をβ簡約を含む最小の同値関係とすると

$M_F =_\beta F\ (M_F)$

すなわち、M_Fは関数Fの不動点

不動点演算子

$Y = \lambda f.\ M_f = \lambda f.\ (\lambda x.\ f\ (x\ x))\ (\lambda x.\ f(x\ x))$とおくと、

$Y\ F =_\beta M_F$ なので、前のページの議論から

$$Y\ F =_\beta F\ (Y\ F)$$

つまり、$Y\ F$ はFの不動点！

Y は任意の関数Fを引数にとり、そのFの不動点を与える関数なので、

不動点演算子

と呼ぶ。

これを使うと再帰が表現可能

再帰関数と不動点

- 再帰関数の例:

fact(n) = if n=0 then 1 else n * fact(n-1)

fact は等式

f = λn. if n=0 then 1 else n * f(n-1)

を満たす関数f

factgen = λf. λn. if n=0 then 1 else n * f(n-1)

とおけば、factは

f = factgen f

を満たすf、つまりfactgen の不動点！

よって不動点演算子Y を用いれば

fact = Y factgen

と書ける

再帰関数の表現（一般の場合）

- 再帰関数定義 f x = e によって定義される関数fは、

  Y (λf.λx.e)

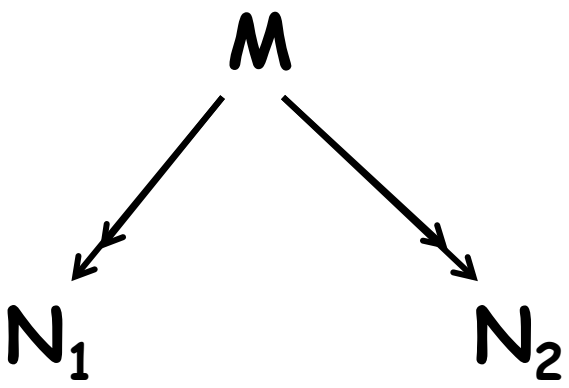
  と（再帰を使わないで）表現可能

本日の内容

- λ計算の表現力（前回の復習）
  - データの表現
  - 不動点演算子と再帰
- λ計算の重要な性質
  - チャーチ・ロッサー性
  - 簡約戦略
- 型付きλ計算

チャーチ・ロッサーの定理

- If

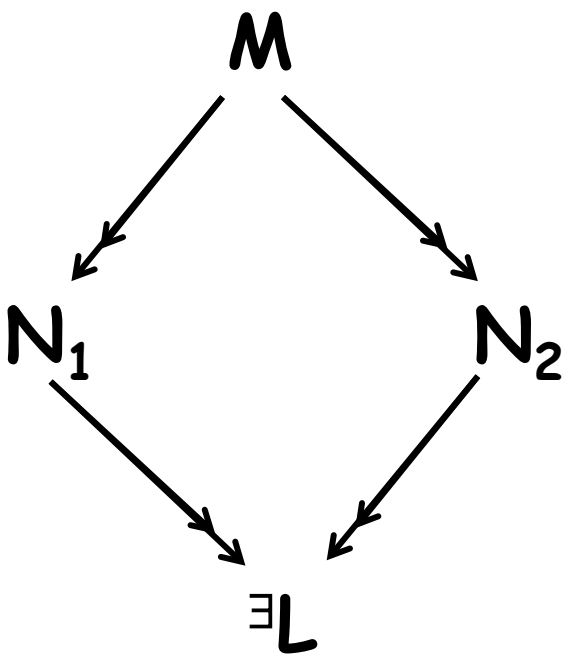


then

$\longrightarrow\!\!\!\!\!\!\rightarrow$ 0ステップ以上の簡約

チャーチ・ロッサーの定理

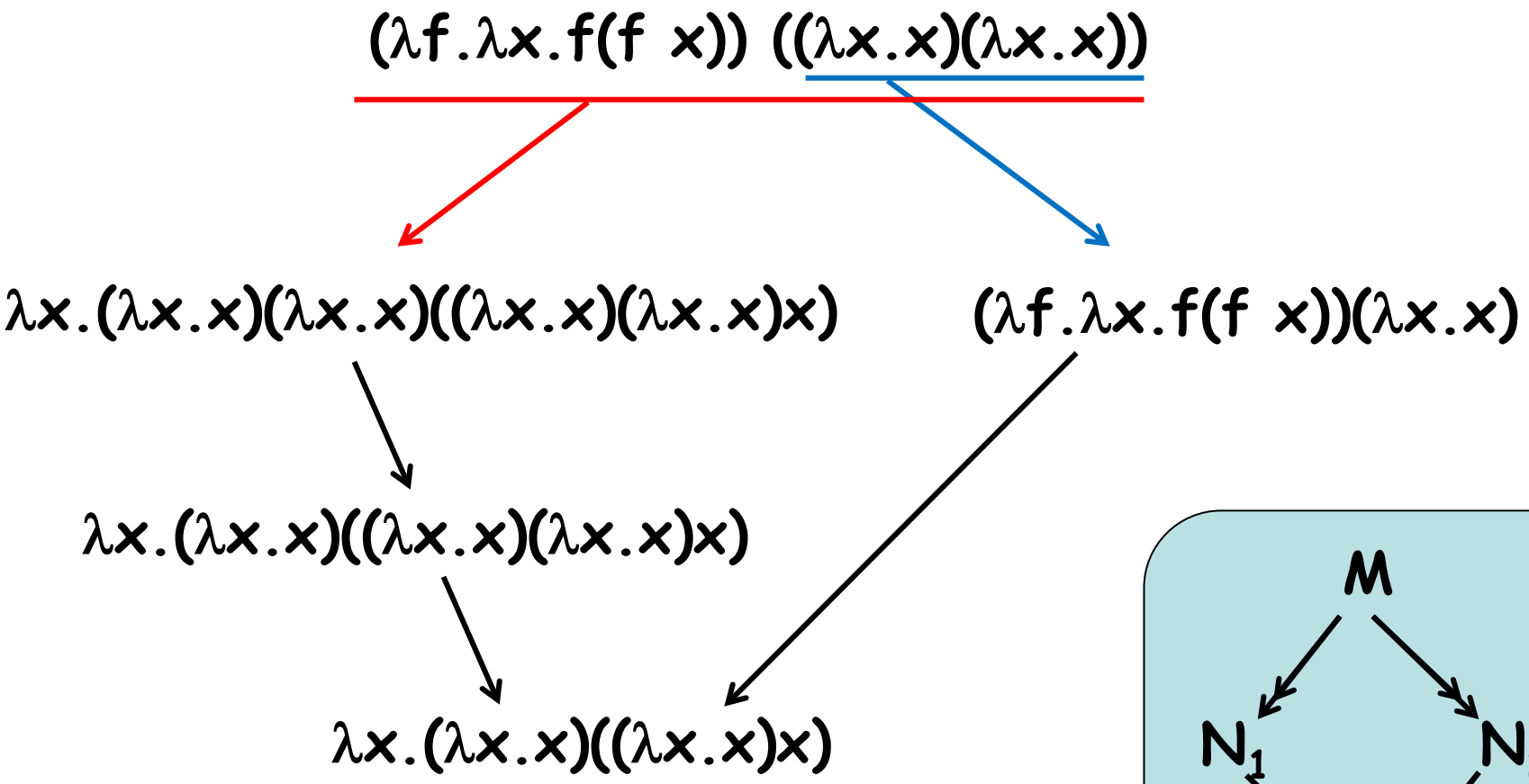
- If

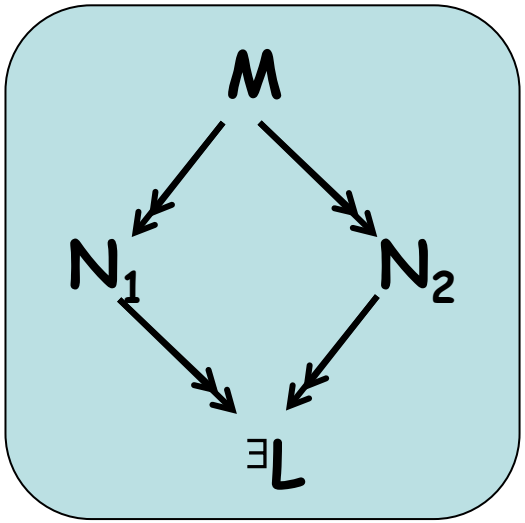


then

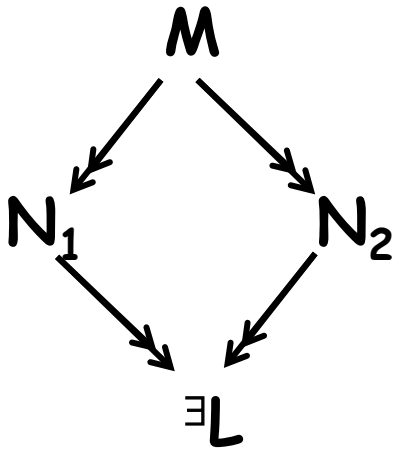
⟶ 0ステップ以上の簡約

例

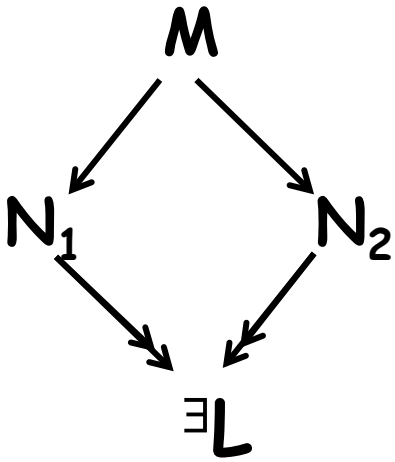

(λf.λx.f(f x)) ((λx.x)(λx.x))
λx.(λx.x)(λx.x)((λx.x)(λx.x)x)
(λf.λx.f(f x))(λx.x)
λx.(λx.x)((λx.x)(λx.x)x)
λx.(λx.x)((λx.x)x)


M
N1
N2
∃L

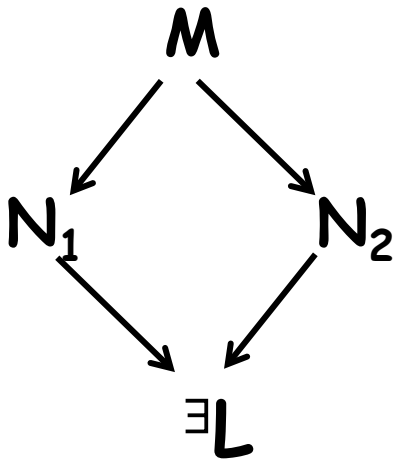
注意



や



は成り立つが、



は成り立たない（前ページの例参照）

⟶ 1ステップの簡約

⟶→ 0ステップ以上の簡約

系：正規系の一意性

定義：$M \rightarrow^* N \not\rightarrow$のとき、NをMの(β)正規形と呼ぶ

定理(チャーチ・ロッサー性の系):
N_1, N_2がMのβ正規形なら（束縛変数の付け替えを除いて）$N_1=N_2$

証明： $M \rightarrow^* N_1 \not\rightarrow$かつ$M \rightarrow^* N_2 \not\rightarrow$と仮定する。

チャーチ・ロッサーの定理より、あるLが存在して

$N_1 \rightarrow^* L$かつ$N_2 \rightarrow^* L$。

ところが、N_1 , N_2ともに簡約できないので

$N_1=L= N_2$

系：正規系の一意性

定義：$M \rightarrow^* N \not\rightarrow$のとき、NをMの(β)正規形と呼ぶ

定理(チャーチ・ロッサー性の系):
N_1, N_2がMのβ正規形なら（束縛変数の付け替えを除いて）$N_1=N_2$

注意：上の定理は、「”簡約が停止するならば”得られる正規形は一意である」ことを述べているだけであり、どのような簡約列を選んでも正規形が得られることは保証しない。
例：$(\lambda x.y)\ ((\lambda x.xx)(\lambda x.xx))$

本日の内容

- λ計算の表現力（前回の復習）
  - データの表現
  - 不動点演算子と再帰
- λ計算の重要な性質
  - チャーチ・ロッサー性
  - 簡約戦略
- 型付きλ計算

簡約戦略

- 簡約の各ステップで、どのβ簡約基（(λx.M)Nの形をした部分項）について簡約を行うかを決める戦略
  - 最左簡約：β簡約基のうち、最も左から始まるものを簡約

e.g.

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

$(\lambda f.\lambda x.f(f\ x))\ ((\lambda x.x)(\lambda x.x))$

- 定理: Mがβ正規形を持つなら、最左簡約によって得られる

本日の内容

- λ計算の表現力（前回の復習）
  - データの表現
  - 不動点演算子と再帰
- λ計算の重要な性質
  - チャーチ・ロッサー性
  - 簡約戦略
- 型付きλ計算

型付きλ計算

- 値の種類を表す「型」を各項に割り当てて、無意味な項を排除
  - ✓ $\lambda x{:}int.x+1 : int \to int$
  - × $(\lambda x{:}int.x)+1 : ??$
- 型付きプログラミング言語の基礎
- 論理学との密接な対応（Curry-Howard同型対応）
- （再帰を加えないかぎり）チューリングマシンと同等の表現力は必ずしも持たない

単純型付きλ計算

- 構文

  τ (型)　　::= b ｜ $\tau_1 \to \tau_2$

  b (基本型) ::= int | bool | ...

  M（項）　　::= $c^{b1 \to \ldots \to bk \to b}$（定数）

  ｜ x ｜ $\lambda x{:}\tau.M$ ｜ $M_1 M_2$

- 簡約

  βの簡約　＋　定数の簡約:

  $c^{b1 \to \ldots \to bk \to b}\ c_1^{b1} \ldots c_k^{bk} \to [c](c_1,\ldots,c_k)$

  (ただし[c] はcが表す数学的な関数。

  例えば [+] 1 2 = 3)

型判断(type judgment)

- $\underbrace{x_1:\tau_1, \ldots, x_n:\tau_n}$ $\vdash M: \tau$

  型環境(変数の有限集合から型集合への写像)

  「型環境$x_1:\tau_1, \ldots, x_n:\tau_n$ のもとで項Mは型τを持つ」

  「各変数x_iが型τ_iの値に束縛されている下でMを評価すると、評価結果はτ型の値になる」

  e.g. $f:\mathrm{int} \to \mathrm{int}, x:\mathrm{int} \vdash f\ x : \mathrm{int}$

  $y:\mathrm{int} \vdash +^{\mathrm{int}\to \mathrm{int}\to\mathrm{int}}\ y\ 1^{\mathrm{int}}: \mathrm{int}$

- 次のスライドの「型付け規則」により定義

型付け規則

$$\Gamma \vdash c^{\tau} : \tau$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

$$\frac{\Gamma, x:\tau_1 \vdash M:\tau_2}{\Gamma \vdash \lambda x:\tau_1 . M : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash M : \tau_1 \to \tau_2 \quad \Gamma \vdash N : \tau_1}{\Gamma \vdash M\ N : \tau_2}$$

型の導出例

- 黒板で

$$\vdash (\lambda f{:}int \to int.\lambda y{:}int.f(f(y)))\ (\lambda z{:}int.\ +^{int\to int\to\ int}\ z\ 1^{int}) : int \to int$$

単純型付きλ計算の性質

- 型の一意性

  Γ |- M:τ かつΓ|-M:τ' ならば$\tau=\tau'$

- 強正規化定理

  Γ |- M:τ ならば無限簡約列$M \to M_1 \to M_2 \to M_3 \to \ldots$

  は存在しない

  => 型なしλ計算やチューリングマシンに比べて表現力が劣る。

- 型保存(subject reduction)

  Γ |- M:τ かつ$M \to N$ ならば Γ |- N:τ

  系: Γ |- M:τ かつ $M \to^* C[(\lambda x{:}\sigma.L)N]$ ならば、Nの型はσ

  （すなわち、関数が要求する引数の型と実際の引数の型は一致する）

単純型付きλ計算の性質

- 型の一意性

  $\Gamma \mid\!- M:\tau$ かつ$\Gamma \mid\!- M:\tau'$ ならば$\tau=\tau'$

- 強正規化定理

  $\Gamma \mid\!- M:\tau$ ならば無限簡約列$M \to M_1 \to M_2 \to M_3 \to \ldots$

  は存在しない

  => 型なしλ計算やチューリングマシンに比べて表現力が劣る。

- 型保存(subject reduction)

  $\Gamma \mid\!- M:\tau$ かつ$M \to N$ ならば $\Gamma \mid\!- N:\tau$

  系: $\Gamma \mid\!- M:\tau$ かつ $M \to^* C[(\lambda x{:}\sigma.L)N]$ ならば、Nの型はσ

  （すなわち、関数が要求する引数の型と実際の引数の型は一致する）

ほとんどの型付き計算モデル、言語で成り立つ性質

単純型付きλ計算の性質

- 型の一意性

  Γ |- M:τ かつΓ|-M:τ' ならば$\tau=\tau'$

- 強正規化定理

  Γ |- M:τ ならば無限簡約列$M \to M_1 \to M_2 \to M_3 \to \ldots$ は存在しない

  => 型なしλ計算やチューリングマシンに比べて表現力が劣る。

- 型保存(subject reduction)

  Γ |- M:τ かつ$M \to N$ ならば Γ |- N:τ

  系: Γ |- M:τ かつ $M \to^* C[(\lambda x{:}\sigma.L)N]$ ならば、Nの型はσ

  （すなわち、関数が要求する引数の型と実際の引数の型は一致する）

一部の型付き計算モデルのみで成り立つ性質

以下のアウトライン

- 型検査アルゴリズム
- 型推論
- 単純型つきλ計算の拡張

型検査アルゴリズム TC

TC: Γ, Mを入力としてΓ |- M:τを満たすτを(あれば)出力

$TC(\Gamma, \mathbf{c}^{\tau}) = \tau$

$TC(\Gamma, \mathbf{x})$ = if x in dom(Γ) then $\Gamma(\mathbf{x})$ else fail

$TC(\Gamma, \lambda\mathbf{x}{:}\tau_1.\mathbf{M}) = \tau_1 \rightarrow TC(\Gamma\{\mathbf{x}{:}\tau_1\}, \mathbf{M})$

$TC(\Gamma, \mathbf{MN})$ = let $\tau_0 = TC(\Gamma, \mathbf{M})$ in
let $\tau_1 = TC(\Gamma, \mathbf{N})$ in
if τ_0 is of the form $\tau_1 \rightarrow \tau_2$ then τ_2
else fail

$$\Gamma \vdash \mathbf{c}^{\tau}:\tau$$

$$\frac{\Gamma(\mathbf{x})=\tau}{\Gamma \vdash \mathbf{x}:\tau}$$

$$\frac{\Gamma, \mathbf{x}:\tau_1 \vdash \mathbf{M}:\tau_2}{\Gamma \vdash \lambda\mathbf{x}:\tau_1.\mathbf{M}: \tau_1 \rightarrow \tau_2}$$

$$\frac{\Gamma \vdash \mathbf{M}: \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash \mathbf{N}: \tau_1}{\Gamma \vdash \mathbf{M}\ \mathbf{N}: \tau_2}$$

以下のアウトライン

- 型検査アルゴリズム
- 型推論
- 単純型つきλ計算の拡張

型推論

(1) 各変数や式に型を表す変数を割り当てる

(2) 型付け規則に従って、型に関する「方程式」をたてる

(3) 型の方程式を解く

型推論の例

let twice f x = f (f x)

$\alpha_f = \alpha_x \rightarrow \alpha_{f\ x}$

α_M: 部分式Mの型

型推論の例

let twice f x = f (f x)

$\alpha_f = \alpha_x \rightarrow \alpha_{f\ x}$

$\alpha_f = \alpha_{f\ x} \rightarrow \alpha_{f(f\ x)}$

α_M: 部分式Mの型

型推論の例

let twice f x = f (f x)

$\alpha_f = \alpha_x \rightarrow \alpha_{f\ x}$

$\alpha_f = \alpha_{f\ x} \rightarrow \alpha_{f(f\ x)}$

$\alpha_{twice} = \alpha_f \rightarrow \alpha_x \rightarrow \alpha_{f(f\ x)}$

α_M: 部分式Mの型

型推論の例

let twice f x = f (f x)

⇓

$\alpha_f = \alpha_x \rightarrow \alpha_{f\ x}$

$\alpha_f = \alpha_{f\ x} \rightarrow \alpha_{f(f\ x)}$

$\alpha_{twice} = \alpha_f \rightarrow \alpha_x \rightarrow \alpha_{f(f\ x)}$

⇓ 型方程式を解く

$\alpha_f = \alpha_x \rightarrow \alpha_x$　　$\alpha_{f\ x} = \alpha_{f(f\ x)} = \alpha_x$

$\alpha_{twice} = (\alpha_x \rightarrow \alpha_x) \rightarrow \alpha_x \rightarrow \alpha_x$

α_M: 部分式Mの型

以下のアウトライン

- 型検査アルゴリズム
- 型推論
- 単純型つきλ計算の拡張

組型を加えた拡張

- 構文

  τ (型)　　::= b ｜ $\tau_1 \to \tau_2$ ｜ $\tau_1 \times \tau_2$

  b (基本型) ::= int | bool | ...

  M（項）　　::= ... ｜ (M_1, M_2) | fst(M) | snd(M)

- 簡約

  ...

  $fst(M_1, M_2) \to M_1$

  $snd(M_1, M_2) \to M_2$

- 型付け

$$\frac{\Gamma \vdash M: \tau_1 \quad \Gamma \vdash N: \tau_2}{\Gamma \vdash (M, N): \tau_1 \times \tau_2}$$

$$\frac{\Gamma \vdash M: \tau_1 \times \tau_2}{\Gamma \vdash fst(M): \tau_1}$$

$$\frac{\Gamma \vdash M: \tau_1 \times \tau_2}{\Gamma \vdash snd(M): \tau_2}$$

直和型を加えた拡張

- 構文

  τ (型)　　::= ... | $\tau_1+\tau_2$

  M（項）　　::= ... | inl(M) | inr(M)

  | case M_0 of inl(x)=>M_1 | inr(y)=>M_2

- 簡約

  case inl(M) of inl(x)=>M_1 | inr(y)=>M_2 $\rightarrow$ [M/x]M_1

  case inr(M) of inl(x)=>M_1 | inr(y)=>M_2 $\rightarrow$ [M/x]M_2

- 型付け

$$\frac{\Gamma \vdash M: \tau_1}{\Gamma \vdash \mathrm{inl}(M): \tau_1+\tau_2}$$

$$\frac{\Gamma \vdash M: \tau_2}{\Gamma \vdash \mathrm{inr}(M): \tau_1+\tau_2}$$

$$\frac{\Gamma \vdash L: \tau_1+\tau_2 \quad \Gamma, x: \tau_1 \vdash M: \tau \quad \Gamma, y: \tau_2 \vdash N: \tau}{\Gamma \vdash \mathrm{case}\ L\ \mathrm{of}\ \mathrm{inl}(x)=>M \mid \mathrm{inr}(y)=>N: \tau}$$

単純型つきλ計算の様々な拡張

- Sysytem F（型つきλ計算の一種）
  - 型も関数の引数に

    τ (型) ::= b | α | $\tau_1 \to \tau_2$ | $\forall\alpha.\tau$

    M ::= x | $\lambda x{:}\tau.M$ | MN | $\Lambda\alpha.M$ | $M[\tau]$
  - 型推論は決定不能
- 再帰型
  - τ (型) ::= b | α | $\tau_1 \to \tau_2$ | $\mu\alpha.\tau$
  - 不動点演算子が表現可能(=>再帰を表現できる)
- 部分型(Subyping)
  - $\sigma < \tau$:「σ 型の値はτ型の値として使える」

  e.g. int < real,　real → int < int → real
- 依存型
  - 型が値に依存できる

    e.g. int array[n] : サイズnの配列の型

    n:int -> int array[n]:　整数nをうけとり、サイズnの配列を返す関数の型