言語モデル論 10/3

参考書

本間 .. 「モデルと表現」

萩谷 .. 「論理と計算のしくみ」　ほとんどのスライドあり

Univ. of Cambridge Computer Laboratory

Lecture course material.

- Semantics of Programming Language
- Types

その他 京大の方たち

集合とデータ型

公理的集合論

構成的な立場

与えられた集合から別の集合をつくる

直積, Cartesian Product

a, b をペアにする.

順序 なし $\{a,b\}=\{b,a\}$

あり $\langle a,b\rangle$.

$\triangleq \{\{a\},\{a,b\}\}$

性質 $\langle a,b\rangle = \langle c,d\rangle \iff a=c \land b=d$

$\{\{a\},\{a,b\}\} = \{\{c\},\{c,d\}\}$

$\iff a=c \land b=d$ を示せ.

$A\times B \triangleq \{\langle a,b\rangle \mid a\in A \land b\in B\}$

$A\times B\times C$ は?

1. $\langle a,\langle b,c\rangle\rangle$
2. $\langle a,\langle b,\langle c,\mathrm{nil}\rangle\rangle\rangle$ cf. リスト
3. $\{\langle 0,a\rangle,\langle 1,b\rangle,\langle 2,c\rangle\}$, cf. array

index

N-tuple $A^2, A^3, \ldots$

A と A^1 は同じか？

cf. 文字 vs 文字列 → 同一視は可.

char　string

A^0 はどうなるか？ → そのような集合の要素の数は1つ (空).

cf. $\{[\,]\}$, $\{nil\}$, 一点集合.

OCaml では unit → 値は ()

Java void

関数とかの void は「何も返さない」のではなくて、一点集合を返す.

f() 何もないのではなくて、「空」がある.
一点集合

対等 (equipollent).

$f: A \to B$, $g: B \to A$

$\forall x \in A\,(g(f(x)) = x) \wedge \forall y \in B\,(f(g(y)) = y)$　　全単射 同値

A と B は <u>対等</u> $A \simeq B$

f, g は一対一対応.

Note 一対一対応 vs 一対一写像
bijection　injection.

直和. disjoint union

cf. OCaml の variant

$A + B \stackrel{\mathrm{def}}{=} \{\langle 0, a\rangle \mid a \in A\} \cup \{\langle 1, b\rangle \mid b \in B\}$

cf. $A \cup B$ とはちがう.

ex. 2分木. $\boxed{0}$ or $\boxed{1 \mid \text{left} \mid \text{right}}$

連絡先 $\boxed{0 \mid \text{e-mail}}$ or $\boxed{1 \mid \text{tel} \mid \quad}$

関数

内包的 $y = x^2+3x-5$

外 〃 $\{\cdots \langle -1,-7\rangle, \cdots\}$

— 関数のグラフ

グラフ関数が等しい関数は等しい。（外延的等しい）

$$f : A \to B$$

domain codomain (range)

を実際にとる値とはちがう。

$A \to B$ は B^A とかく。

$\{\}^{\{\}}$ はどんな集合?

$\{\} \to \{\}$?

$= \{\underline{\underline{\{\}}}\}$.

$|\{\}^{\{\}}| = 1$

cf. $0^0 = 1$

$\{0,1\} \to \{0,1\}$.

一方，$\{\}^{\{1,2,3\}} = \{1,2,3\} \to \{\}$

は0通り。

cf $0^3 = 0$.

論理モデル論.

Scala での例.

Set(1,3,4,6,8)(6) → true

" (7) → false

Map("Japan" → "Tokyo", "France" → "Paris")("Japan") → Tokyo.

Rev.

1^0 はどんな系? → equiv. to $0^0 = 1$

関数.

$f \in B^A \Leftrightarrow$ 1. $f \subseteq A \times B$

2. $\wedge\ \forall x \in A.\ \exists y \in B.\ (\langle x, y\rangle \in f)$.

3. $\wedge\ \forall x \in A\ \forall y \in B\ \forall y' \in B$

$(\langle x, y\rangle \in f \wedge \langle x, y'\rangle \in f \Rightarrow y = y'$

ないと. f は partial

Currying

f(x, y) とは?

1. $f'(\langle x, y\rangle)$ ととらえる → $A \times B \to C$

2. $f''(x)(y)$ ととらえる. → $A \to (B \to C)$.

論理では?

$P \wedge Q \Rightarrow R$

$P \Rightarrow (Q \Rightarrow R)$.

0引数関数?

SML# の例.

\# pwd;

unit → string

\# pwd ();

→ 'value'

→ 値にしないで, 関数にしておきたい.

- 実行に時間がかかるとき. 計算のタイミング
- 副作用

n 要素の一次元配列. A^n

$\{0,\dots,n-1\} \to A$ 型の関数

は 同等

べき集合 2^S. powerset.

$2^{\{\}}$ は? → ちょうど要素が1個ある集合.

equiv. to $2^0 = 1$

2^A と $A \to \{0,1\}$ は同等.

S の特性関数.

多重集合. multiset

関係.

$\{\langle x,y\rangle \mid$ x と y 間に なりたつ 関係 $\}$.

:

1 項関係は その部分集合.

0 項関係とは何か?

...

$R \subseteq A^2$ ← 2 項関係.

$R \subseteq 2^A$ ← 1 項関係.

$R \subseteq \underbrace{2^{A^0}}_{= \{\{\}, \{\{\}\}\}}$, $A^0 = \{\{\}\}$ ← 0 項関係.

equiv. boolean

集合族. (Indexed) family of sets

添字付き

$\{A_i\}_{i\in I}$.

$\{\langle i, A_i\rangle \mid i \in I\}$

族の Π.

int[] a = new int[100].

$\mathrm{Int}(\mathbb{N}) = \mathrm{Int} \times \cdots \mathrm{int}$

$A_i \leftrightarrow \mathrm{int}^{I}$

$A := \{A_i\}_{i \in I}$.

$B := \bigcup_{i \in I} = \bigcup \{A_i \mid i \in I\}$.

$\prod A = \prod_{i \in I} A_i \overset{*}{=} \{ f \in B^I \mid \forall i \in I\ (f(i) \in A_i) \}$

◦ 余題.

$I = \{0, 1, 2\}$ で $\prod_{i \in I} A_i$ は?

Q. $\prod_{i \in \{0,1,2\}} A_i \cong A_0 \times A_1 \times A_2$. となってほしい.

Q. $\prod_{i \in I} A \cong A^I \quad (I \to A)$. であるべき

族の Σ.

直和の一般化.

$\sum A = \sum_{i \in I} A_i = \{ \langle i, x \rangle \mid i \in I \wedge x \in A_i \}$

◦ $\sum_{i \in \{0,1,2\}} A_i \cong A_0 + A_2 + A_3$

◦ $\sum_{i \in I} A \cong I \times A$.

再帰的に定義される集合.

(1). 帰納的データ構造

例. Aの要素を非終端節点にもつ
2分木の集合 T

2分木は (i) 空であるか ・でなく (ii) 節点(根) が2つの部分木をしたがえたもの である.

$$T_0 \triangleq \{\}$$

$$T_{n+1} \triangleq \{\langle 0, \{\cdot\}\rangle\} \overset{\text{Tag}}{\cup} \{\langle 1, \langle a, x, y\rangle\rangle \mid a \in A,\ x \in T_n,\ y \in T_n\}$$

$I_A(S) \triangleq \{\cdot\} + A \times S \times S$ とおく.

$$T_{n+1} \triangleq I_A(T_n).$$

$$T \triangleq \bigcup_{n \in N} T_n = \bigcup_{n \in N} I_A{}^n(\{\}).$$

$$T_0 = \{\}.$$

$T_1 = \{\cdot\} + \underbrace{A \times T_0 \times T_0}_{\{\}}$ 高さ1未満の2分木全体

$T_2 = \{\cdot\} + A \times T_1 \times T_1$ 高さ2 ‥

⋮

言語モデル論 11/7

コンフィギュレーション (計算の状態)
の集合 Config と その間の遷移関係

集合の帰納的定義
形式体系スタイル

(公理) c　　　　c は集合 S に属する

(推論規則) $\dfrac{h_1\ h_2\ \cdots\ h_n}{c}$　　　　$h_1 \cdots h_n$ が S に属するならば、c も S に属する

集合 S に属する
$\Leftrightarrow$ 公理と推論規則を有限回
組み合わせて証明できる

コンフィギュレーション

Phrase $\stackrel{\text{def}}{=}$ プログラム句の集合

States $\stackrel{\text{def}}{=}$ $\underset{\text{記憶域}}{L} \rightharpoonup Z$

location の状態の集合. (部分関数)

cf. state は store
environment など

Config $\stackrel{\text{def}}{=}$ $\underset{\text{プログラム}}{\text{Phrases}} \times \underset{\text{メモリ}}{\text{States}}$

cf. プログラム カウンタ.

dom : States $\to 2^L$

値の入っている location の集合

$$s[l \mapsto n](l') \stackrel{\text{def}}{=} \begin{cases} n & \text{if } l' = l \\ s(l') & \text{if } l' \neq l \wedge l' \in \mathrm{dom}(s). \end{cases}$$

$\to\ \subseteq$ Config $\times$ Config

遷移関係

$$\xrightarrow[(\mathrm{op1})]{} \quad \frac{\langle E_1, s\rangle \to \langle E_1', s'\rangle}{\langle E_1 \ \mathrm{op}\ E_2, s\rangle \to \langle E_1' \ \mathrm{op}\ E_2, s'\rangle}$$

$\downarrow$ bottom up　$\uparrow$ top down

いえる

~~topdownは示して.~~

副作用するものって何, 文だけ.

$$\xrightarrow[(\mathrm{op2})]{} \quad \frac{\langle E_2, s\rangle \to \langle E_2', s'\rangle}{\langle n_1 \ \mathrm{op}\ E_2, s\rangle \to \langle n_1 \ \mathrm{op}\ E_2', s'\rangle}$$

& 左が実行できなくなったら, 右を.

$$\xrightarrow[(\mathrm{op3})]{} \quad \langle n_1 \ \mathrm{op}\ n_2, s\rangle \to \langle c, s\rangle \quad \text{if } c = n_1 \ \mathrm{op}\ n_2.$$

& すべて実行できなくなったら, 実際に行う.

$$\xrightarrow[(\mathrm{loc})]{} \quad \langle !\ell, s\rangle \to \langle n, s\rangle \quad \text{if } \ell \in \mathrm{dom}(s) \ \& \ s(\ell) = n.$$

cf. $\underline{(((1+2)+4)+(3+4))}_{\mathrm{op}}$

代入と逐次実行

$$\xrightarrow[(\mathrm{ass1})]{} \quad \frac{\langle E, s\rangle \to \langle E', s'\rangle}{\langle \ell := E, s\rangle \to \langle \ell := E', s'\rangle}.$$

$$\xrightarrow[(\mathrm{ass2})]{} \quad \langle \ell := n, s\rangle \to \langle \underbrace{\mathrm{skip}}_{\text{文}}, \underbrace{s[\ell \mapsto n]}_{n\text{の}\ell\text{への代入}}\rangle.$$

$$\xrightarrow[(\mathrm{seq1})]{} \quad \frac{\langle C_1, s\rangle \to \langle C_1', s'\rangle}{\langle C_1; C_2, s\rangle \to \langle C_1'; C_2, s'\rangle}$$

$$\xrightarrow[(\mathrm{seq2})]{} \quad \langle \mathrm{skip}; C, s\rangle \to \langle C, s\rangle.$$

if, while

$$(\overrightarrow{\text{if1}})\quad \frac{\langle B, s\rangle \to \langle B', s'\rangle}{\langle \text{if } B \text{ then } C_1 \text{ else } C_2, s\rangle \to \langle \text{if } B' \text{ then } C_1 \text{ else } C_2, s'\rangle}$$

$$(\overrightarrow{\text{if2}})\quad \langle \text{if true then } C_1 \text{ else } C_2, s\rangle \to \langle C_1, s\rangle$$

$$(\overrightarrow{\text{if3}})\quad \langle \text{if false then } C_1 \text{ else } C_2, s\rangle \to \langle C_2, s\rangle$$

$$(\overrightarrow{\text{whl}})\quad \langle \text{while } B \text{ do } C, s\rangle \to \langle \text{if } B \text{ then } (C; \text{while } B \text{ do } C) \text{ else skip}, s\rangle$$

cf. in Scala

```
def ~~mywhile~~ myWhile ( b: => Boolean ) ( c: => Unit ) {
    if (b) { c; myWhile(b)(c) } else ()
}
```

仮引数に ⇒ をつけて。
call by name にできる。

call by value. 評価して、値を渡す
call by name. 評価せず、式を渡す。
使うときに評価する

⇒ Boolean
is Unit ⇒ Boolean.

c: ⇒ Unit
評価する式.

ex. myWhile (k>0) { s = s+k; k = k-1; }

C 言語の do, while, for を定義できる.

LC プログラムの実行の行方

a. きちんと止まる (terminate)

b. 行きづまる (stuck)

c. 止まらない (diverge)

○ termination.

いずれか。

$\langle n, s\rangle$ $\langle \mathrm{true}, s\rangle$ $\langle \mathrm{false}, s\rangle$ $\langle \mathrm{skip}, s\rangle$

コンフィギュレーションが最終でなくて（上の4つ）
遷移不可能なとき、stuck という。

ex. $\langle (!l+1), \{l' \mapsto 1\}\rangle$

は $l \neq l'$ なので stuck

代入されていない loc の値は読むことができない。

ex.

$$\langle c, s\rangle \to \langle \mathrm{if}\ B\ \mathrm{then}\ (c';c)\ \mathrm{else}\ \mathrm{skip}, s\rangle$$
$$\to \langle \mathrm{if}\ 4>0\ \mathrm{then}\ (c';c)\ \mathrm{else}\ \mathrm{skip}, s\rangle$$
$$\to \langle \mathrm{if}\ \mathrm{true}\ \mathrm{then}\ (c';c)\ \mathrm{else}\ \mathrm{skip}, s\rangle$$
$$\to \langle c';c, s\rangle$$
$$\xrightarrow{\text{たくさん}}{}^{*} \langle \mathrm{skip}, \underbrace{s[l \mapsto 0, l' \mapsto 24]}\rangle$$

最終的に $l = 0$, $l' = 24$ になる。

⋮

c は c' を 5回プリント.

ex.

$\langle \text{if } !l>0 \text{ then } (c';c) \text{ else skip}, \{l\mapsto 4, l'\mapsto l\}\rangle$

$\quad \to \langle \text{if } 4>0 \text{ then } (c';c) \text{ else skip}; \{l\mapsto 4, l'\mapsto 1\}\rangle$

は以下のように証明

$$\dfrac{\dfrac{\dfrac{}{\langle !l, \{l\mapsto 4, l'\mapsto 1\}\rangle \to \langle 4, \{l\mapsto 4, l'\mapsto 1\}\rangle}\ (\text{loc})}{\langle !l>0, \{l\mapsto 4, l'\mapsto 1\}\rangle \to \langle 4>0, \{l\mapsto 4, l'\mapsto 1\}\rangle}\ (\text{op1})}{}$$

LCの意味論の性質を証明

1. 決定性

   $\langle P, s\rangle \to \langle P', s'\rangle$ かつ $\langle P, s\rangle \to \langle P'', s''\rangle$

   ならば, $P' = P''$ かつ $s' = s''$

2. Subject reduction による型の保存

   $\langle P, s\rangle \to \langle P', s'\rangle$ ならば.

   P, P' は同じ型をもつ.

3. 式の副作用がない.

   ~~式の型~~

   P が式のとき, $\langle P, s\rangle \to \langle P', s'\rangle$

   ならば $s = s'$.

プログラム文 P の構造に関する. 帰納法

決定的 自明でない例.
(自明でない証明も多い)

$(1+3) + (5+7)$

$\swarrow \qquad \searrow$

$4 + (5+7) \qquad (1+3) + 12$

$\searrow \quad \swarrow$

$4 + 12$

$\downarrow$

16.

問題.

small-step semantics に
どのようなルールを加えたらいいか

- ++
- while(1)

  トップレベルで fork

Natural (big-step) semantics

◦ For given $\langle p_0, s_0 \rangle$
止まっても、

$$\langle p_0, s_0 \rangle \to \cdots \to \langle p_n, s_n \rangle \not\to$$ 次の実行がない。

なる実行列 ($n \geq 0$) が <u>一意に定まる</u>

◦ 実行では $\langle p_0, s_0 \rangle$ が <u>最後どうなるか</u> に興味がある

$\langle p, s \rangle$, ~~last~~ config $\underset{\text{value}}{\underset{=}{\langle v, s' \rangle}}$.

その関係 ($\Downarrow$) を直接定義したい。

$$\langle p, s \rangle \Downarrow \langle v, s' \rangle$$

注. ◦ 最後に至らない計算は扱わない。
(発散と行き詰まりは扱わない)

$loc(p)$ を p に現れる location の集合として、
$loc(p) \subseteq dom(s)$ ならば、行き詰まりを起こさない。

Prove $\langle p, s \rangle \to \langle p', s' \rangle$ かつ $loc(p) \subseteq dom(s)$,
$\to loc(p') \subseteq dom(s')$

一回の計算のステップに内在する保存則.

発散するかどうかは 決定不能

他ルールはプリントで見よ.

・small-step semantics との関係

$\forall$ configuration $\langle P, s\rangle$,

$\forall$ terminal config $\langle V, s'\rangle$.

$$\langle P, s\rangle \Downarrow \langle V, s'\rangle \iff \langle P, s\rangle \to^{*} \langle V, s'\rangle$$

が成り立つ.

ブロック構造と局所状態の導入. 　　　　LCの7章だって.

Phrases : $P ::= C \mid E \mid B$

Commands :

$$\begin{aligned} C ::= {} & \mathrm{skip} \mid l := E \mid C ; C \mid \\ & \mathrm{if} \, .. \mid \mathrm{while} \, .. \mid \\ & \underline{\mathrm{begin\ loc}\ l := E ; C\ \mathrm{end}.} \end{aligned}$$

これをadd.

big step sem. はこうなる.

$$(\Downarrow_{ls}) \quad \frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle. \quad \langle C[l'/l], s'[l' \mapsto n]\rangle \Downarrow \langle \mathrm{skip}, s''[l' \mapsto n']\rangle}{\langle \mathrm{begin\ loc}\ l := E ; C\ \mathrm{end}, s\rangle \Downarrow \langle \mathrm{skip}, \underline{s''}\rangle}$$

ダブっていないものなら何でもよい

l'の初期値として, n (Eの値) を入れる.

実行したものの中で $l' \mapsto n'$ をのぞいたもの.

応用

プログラムの等価性の証明.

で効果的には

- 何と値 (ひとつ?)
- 止まらないときは何か.

cf. ステップ数.

$P_1 \simeq P_2$ とは、すべての s, v, s' について.

$$\langle P_1, s\rangle \Downarrow \langle \underbrace{v}_{同じ値}, \underbrace{s'}_{同じメモリ} \rangle \iff \langle P_2, s\rangle \Downarrow \langle \underbrace{v}_{同じ値}, \underbrace{s'}_{同じメモリ} \rangle.$$

ex. 例.

- $(\text{if } B \text{ then } C \text{ else } C'); C''$
  $\simeq (\text{if } B \text{ then } (C; C'') \text{ else } (C'; C''))$
- $(C; C'); C'' \simeq C; (C'; C'')$

一方ずつ 示すのみ.

- $C_1 \simeq C_2 \Rightarrow \text{while } B \text{ do } C_1 \simeq \text{while } B \text{ do } C_2$

  ∵ 実行ステップに関する数学的帰納法
  1ステップにバラして、すべてのステップ数に
  関する数学的帰納法を用いる

言語モデル論 11/14

The lambda calculus

cf. Foundation of Functional Programming.
Lawrence C. Paulson and Alan Mycroft

First-class / Second-class.

変数や式の値に自由になれるもの

多くの命令言語のうちに限定されている

First-class citizen.

- 実行時に作成できる
- 変数に代入できる
- 手続きや関数の引数になれる
- 関数の戻り値になれる

例. 関数の定義.

$f(x) = x^2 + 2ax + a^2$ ← 定義

$f(a) = 4a^2$ ← 適用.

定義と適用の区別がつかない.

$f = \lambda x.\ x^2 + 2ax + a^2$

$f\ a = 4a^2$

できる

例. 不定積分 (原始関数).

$\int x^2 dx = \frac{x^3}{3} + C$ ← ちがうもの!?

cf. $\int y^2 dy = \frac{y^3}{3} + C$

ラムダ式.

$\int (\lambda x.x^2) = \lambda x.\left(\frac{x^3}{3} + C\right)$ ← 同じものであることがはっきりする

$\int (\lambda y.y^2) = \lambda y\ \left(\frac{y^3}{3} + C\right)$

定積分.

$F = \int (\lambda x.\ x^2)$ とおいて.

$$\int_a^b (\lambda x. x^2) = F(b) - F(a).$$

$$\int_a^b f = (\int f)\, b - (\int f)\, a$$ つまり

$$\int_\alpha = \lambda f. \lambda x. \lambda y = \int f\, y - \int f\, x$$

例. 微分.

何に関して微分しているか分からない.

$$f^{①}(x) = 2x + 2a.$$

$$f'(a) = 4a$$

$$\rightsquigarrow \quad f' = (\lambda x.\ x^2 + 2ax + a^2)' = \lambda x.\ 2x + 2a$$

$$f'\, a = 4a$$

例 和.

$$\sum_{k \in S} f(k)$$

$$\rightsquigarrow \quad \sum_S f$$

例. 述語論理

$$\exists y.\ \mathrm{likes}(x, y)$$

$$\rightsquigarrow \quad \exists(\lambda y.\ \mathrm{likes}(x, y))$$

: ∃は高階関数.

$$\forall x\, \exists y.\ \mathrm{likes}(x, y)$$

$$\rightsquigarrow \quad \forall(\lambda x. \exists(\lambda y. \mathrm{likes}(x, y)))$$

2.4. 自由変数と束縛変数

束縛変数：「仮の名前」、変更可能.

自由変数：外から見える、勝手に名前を変えられない.

$FV(M)$：λ式 M の外から見える変数の集合

$$FV(x) = \{x\}$$

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

xを除く

$$FV(MN) = FV(M) \cup FV(N).$$

$FV(M) = \{\}$ のとき、

Mは閉じたλ式である。

または結合子（combinator）という。

2.5. λ式の簡約（reduction）

2.5.1 λ計算の small-step semantics.

中心はβ簡約（β-reduction）.

$$(\beta) \quad (\lambda x.M)N \to M[x := N]$$

：M中の自由変数 x を Nに置きかえ.

メタ記号

推論規則

$$\frac{M \to M'}{MN \to M'N} \qquad \frac{M \to M'}{LM \to LM'} \qquad \frac{M \to M'}{\lambda x.M \to \lambda x.M'}$$

非決定的、さまざまな評価戦略がある

2.3 λ式の定義.

アルファベット (記号).

$x, y, z \ldots \in Var$ (可算無限個).

λ.

λ式の構文.

$$
\begin{aligned}
M ::= &\ x \\
|\ &(\lambda x. M) \quad \text{abstraction 抽象} \\
|\ &(M\ M) \quad \text{application 適用}
\end{aligned}
$$

λ式の全体を Λ で書く.

記法.

L, M, N λ式.

$M \equiv N$ 同じ形をしている

$M = N$ 等価である

略記法.

$(M N)$ は $M N$ で書いてよい.

LMN は $(LM)N$ のこと.

$\lambda x. MN$ は $\lambda x (MN)$ のこと. (最長一致).

$\lambda x_1, x_2, \ldots, x_n. M$ は

$\lambda x_1. \lambda x_2. \ldots \lambda x_n. M$ のこと

省略化.

例.

x

$(xz)(yz)$

xx 自己適用.

$\lambda x. \lambda y. \underline{f}(x, y)$

定数.

$\lambda x. x$ 恒等関数 (I).

$\lambda x. y$ 定数 〃

例. $\underline{(\lambda x.\ \underline{(\lambda y.\ xy)\ z})\ w}$

$(\lambda x.\ xz)\ w$　　$(\lambda y.\ wy)\ z$　　どちらでもいい

$w\ z$

2.5.2.

評価の行く末.

(1) いずれ簡約できなくなる

— 正規形. (normal form)

(2) 簡約が無限につづく.

ex.

$(\lambda x.\ x)\ a$ は 正規形 でないが,

正規形 を 持つ.

$\Omega \equiv (\lambda x.\ xx)(\lambda x.\ xx)$ は 正規形をもたない.

$\because (\Omega \to \Omega)$.

(a) 評価戦略によらずに (1)

(b) 〃 よって (1 or 2).

(c) 〃 によらずに (2)

正規形をもつ場合.
- それは 1通りか?
- 必ず正規形に至る戦略は あるのか?

7.5.3 λβの等しさ.

$\to$ は 2項関係だが, 同値関係ではない.

直感的には $M=N$ とは 0回以上の reduction と

その逆操作 (expansion) で, M を N に 変形.

$$\begin{tikzcd}[column sep=small]
M \arrow[dr, "*"] & & M_1 \arrow[dl, "*"'] \arrow[dr, "*"] & & M_2 \arrow[dl] \arrow[dr] & \cdots & & M_{n-1} \arrow[dl] \arrow[dr, "*"] & & N \arrow[dl, "*"'] \\
& L_1 & & L_2 & & & \ & & L_n &
\end{tikzcd}$$

$\xrightarrow{*}$ は $\to$ の 反射推移閉包

$=$ は $\to$ の 〃 対称閉包

$$(\to \cup \leftarrow)^*.$$

3.5.3 λ式の等しさ

(前回) $\rightarrow$

$=$ は $(\rightarrow \cup \leftarrow)^*$

問. $(\rightarrow \cup \leftarrow)^*$ が $\rightarrow$ を含む最小の同値関係であることを示せ.

$=$ は λ式の構成法に閉じた合同関係である.
(congruence)

$$\frac{M = M'}{MN = M'N} \qquad \frac{M = M'}{LM = LM'} \qquad \frac{M = M'}{\lambda x M = \lambda x M'}$$

◎ 2つのλ式が等しいかどうかは決定可能か?

◎ すべてのλ式が=になってしまうことはないか.

3.5.4. η-簡約 と η-展開. (η-変換).
(eta-reduction) (eta-expansion).

$(\eta) \quad (\lambda x.\, Mx) \leftrightarrow M \ \text{ if } x \notin FV(M)$.

関数の外延的な等しさを表現
(実行らしくはない)

η変換の有無は, λ計算の本質部分には影響しない.

3.6 代入の定義と α同値.

$M[x:=N]$ の定義は自明ではない.

例). $(\lambda x.yx)[y:=z] = (\lambda x.zx)$ ok.

$(\lambda z.yz)[y:=z] = (\lambda z.zz)$. 自由変数の捕捉がおこる.

このzはこれ.

このzは外のz.

定義.

$$x[y:=L] \triangleq \begin{cases} L & \text{if } x=y \\ x & \text{if } x\neq y. \end{cases}$$

$$(MN)[y:=L] \triangleq (M[y:=L])(N[y:=L]).$$

Lにxがつかまれない.

$$(\lambda x.M)[y:=L] = \begin{cases} \lambda x.M & \text{if } x=y \\ \lambda x.(M[y:=L]) & \text{if } x\neq y \wedge x\notin FV(L) \\ \lambda z.(M[x:=z])[y:=L] & \text{if } x\neq y \wedge x\in FV(L) \wedge \underset{\sim}{z} \notin FV(M)\cup FV(L) \end{cases}$$

freshな変数z.

α変換 (α-conversion).

(α) $\lambda x.M \equiv \lambda y.M[x:=y]$ if $y\notin FV(M)$.

計算の観点からは本質的でない.

今後はα同値なλ式は同一とみなす.

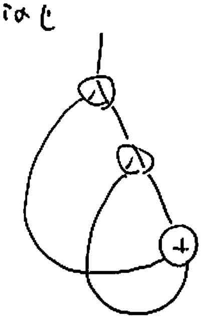
cf. Bruijn 記法.

いくつ外側のλ抽象の束縛変数を指しているかを番号で示す.

ex. $\lambda x.\lambda y.(+x)y$ は

$\lambda\lambda(+1)0$ となる

自然数で代用し
自然でない



別の表示.

λの計算 (1990年頃)

ラムダ式を用いて
さまざまなものを表現する.

(a) ブール値と条件分岐.

$\overline{\text{if}}$ は ほとんど何もしてない.

(b) 順序対.

$\overline{\text{pair}} \triangleq \lambda xy.\lambda p.\ pxy$

ex. $(\overline{\text{pair}}\ M\ N)\ \overline{\text{true}} \to^{*} M.$

$\overline{\text{false}} \to^{*} N.$

順序対を入れ子にすれば 定義し

リストも.

(c) 自然数とその表現

$\overline{0} \triangleq \lambda fx.x$

$\overline{1} \triangleq \lambda fx.fx.$

$\overline{2} \triangleq \lambda fx.f(fx).$

$\vdots$

点の部分の中に.

(d) 自然数とその演算.

$\overline{\text{add}} \triangleq \lambda mn.\ \lambda fx.\ nf\,(mfx)$

$\overline{\text{mul}} \triangleq \lambda mn.\ \lambda fx.\ n(mf)x.$

自明ではない

$\overline{\text{exp}} \triangleq \lambda mn.\ nm$

m^{n}

これで 0^0 をやってみる. 0になる?
1になる?

$\overline{\text{iszero}} \triangleq \dots$

$\overline{\text{pre}} \triangleq \lambda n.\lambda fx.\ \text{snd}\ (n(\overline{\text{pair}} f)(\overline{\text{pair}}\ x\ x))$

: predesessor fun.

3.8 不動点 組合子

一般に x が f の 不動点 $\triangleq$ $f(x) = x$.

(fix point, fixed point)

実数関数の場合.

$f = \lambda x.x^2$ 複数の不動点

$f = \lambda x.x+1$. 不動点 なし.

$f = \lambda x.x$ 無数の不動点

不動点定理

$$\forall F \in \Lambda .\ \exists X \in \Lambda\ (FX = X)$$

∵) $X \triangleq (\lambda x.F(x\,x))(\lambda x.F(x\,x))$

とすれば, $X \twoheadrightarrow FX$ となる.

$Y \triangleq \lambda f.(\lambda x.f(x\,x))(\lambda x.f(x\,x))$ を

Curry の 不動点 結合子 という.

再帰関数の定義.

$$\overline{fact} \triangleq Y(\boxed{\lambda g.\ \overline{if}\ (\overline{iszero}\ n)\ \bar{1}\ (\overline{mul}\ n\ (\boxed{g}\ (\overline{pre}\ n))))}\ = I\ とおく.$$

穴が全体と
同じになれば循環になる

ふつうは
ここで fact.

g を それら. を下に渡してやる

$\Phi = \lambda n.\bot$ とする ($\bot$: undefined)

Φを部分関数と考えて、$[\![\Phi]\!]$ (Φのグラフ) は空集合.

$\Phi, \mathcal{F}(\Phi), \mathcal{F}^2(\Phi), \ldots$ は ! への近似列.

$[\![\Phi]\!], [\![\mathcal{F}(\Phi)]\!], [\![\mathcal{F}^2(\Phi)]\!], \ldots$ を考えて.

(1) $[\![F]\!] \subseteq [\![G]\!] \Rightarrow [\![\mathcal{F}(F)]\!] \subseteq [\![\mathcal{F}(G)]\!]$.

(2) $[\![\mathcal{F}^n(\Phi)]\!] \subseteq [\![\mathcal{F}^{n+1}(\Phi)]\!]$

↑ より多くのものに対して答えを返してくれる.

(3) $[\![Y\mathcal{F}]\!] = [\![\bigcup_{n\in\mathbb{N}} \mathcal{F}^n(\Phi)]\!] = \;!$

階乗関数.

(4) $[\![Y\mathcal{F}]\!]$ は $[\![\mathcal{F}]\!]$ の最小不動点

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

$\to \lambda x.\ x$.

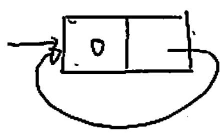
$\lambda f x.\ f x \to$ eta conversion にて $\lambda f.\ f$.

→ 等しい.

より 1 になる.

(iv) $Y\ \overline{\mathrm{succ}}\ fx \xrightarrow{*}$ (III) ?

無限データ構造.



$\overline{\mathrm{zeros}} \triangleq Y(\lambda g.\ \overline{\mathrm{pair}}\ \overline{0}\ g)$.

- Curry の Y 以外にも, 不動点結合子は無数に作れる

$\Theta \triangleq (\lambda x g.\ g(x x g))(\lambda x g.\ g(x x g))$

Turing の不動点結合子.　　前回までにわかってる.

問. $\Theta F \xrightarrow{*} F(\Theta F)$ を確かめよ.

(cf. $YF \xrightarrow{*} F(YF)$ ではない)

相互再帰関数 (f, g)

$$\begin{cases} f = P\,f\,g \\ g = Q\,f\,g \end{cases}$$

を定義するには、$F\underbrace{(f,g)}_{f \text{と} g \text{のペア}} = (P f g,\ Q f g)$ となる F.

の不動点を求める.

$$L \stackrel{\triangle}{=} Y\,(\underline{\lambda p.\ \overline{\mathrm{pair}}\,(P\,(\overline{\mathrm{fst}}\,p)(\overline{\mathrm{snd}}\,p))\,(Q\,(\overline{\mathrm{fst}}\,p)(\overline{\mathrm{snd}}\,p))})\ ⊛.$$

$$⊛\ L = \overline{\mathrm{pair}}\,(P\,(\overline{\mathrm{fst}}\,L)(\overline{\mathrm{snd}}\,L))\,(Q\,(\overline{\mathrm{fst}}\,L)(\overline{\mathrm{snd}}\,L))$$

$$\overline{\mathrm{fst}}\,L = P\,(\overline{\mathrm{fst}}\,L)\,(\overline{\mathrm{snd}}\,L).$$

$$\overline{\mathrm{snd}}\,L = Q\,(\overline{\mathrm{fst}}\,L)\,(\overline{\mathrm{snd}}\,L)$$

$f \qquad g$

<u>paradoxical combinator</u> ∠ Church のもくろみ

$x \in S$	$\leftrightarrow$	Sx	(application)
$\{x \mid P\}$	$\leftrightarrow$	$\lambda x.P$	(abstraction)

とすると.

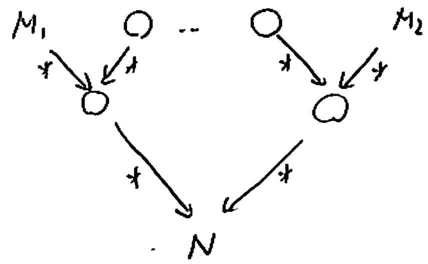
$R = \{x \mid x \notin x\}$	$\leftrightarrow$	$\underline{R = \lambda x.\ \mathrm{not}\ (x\,x)}$.	これがうまくいったら 論理体系としては まずい.
$R \in R$	$\leftrightarrow$	$RR \rightarrow \mathrm{not}\ (RR)$.	

λ計算は論理体系ではなく計算体系として生き残る

3.7 Church-Rosser 定理

$M_1 = M_2$ ならば $M_1 \xrightarrow{*} N$, $M_2 \xrightarrow{*} N$

が存在する.



系. $M = N$ かつ $N \not\to$ ならば $M \xrightarrow{*} N$

これ以上簡約できない.(正規形)

$M = N$ かつ $M \not\to$ かつ $N \not\to$ ならば $M \equiv N$
(up to α変換)

: α変換は自由にやってよい.

◎ 大切. $M \not\to$ かつ $N \not\to$ かつ $M \not\equiv N$ ならば $M \neq N$.

例. $\lambda xy.x\,(\overline{\text{true}}) \neq \lambda xy.y\,(\overline{\text{false}})$.

λ計算は等式理論として無矛盾

$x \neq_\beta \lambda y.xy$

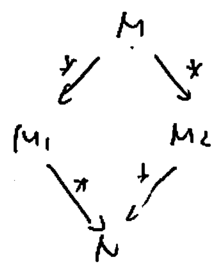
β変換だけでは等しいといえない.

よって η変換は冗長でない.

Diamond Property

$M_1 \xleftarrow{*} M$, $M \xrightarrow{*} M_2$ ならば
$M_1 \xrightarrow{*} N$ かつ $M_2 \xrightarrow{*} N$ なる N が存在

ex.



これは Church-Rosser property ということもある

注

$$\begin{array}{ccc} & M & \\ \swarrow & & \searrow \\ M_1 & & M_2 \\ \searrow & & \swarrow \\ & N & \end{array}$$

(small diamond property) はなりたたない.

正規形は
ユニークである

ex. $(\lambda x.xx)((\lambda x.x)a)$

$$\begin{array}{ccc} & (\lambda x.xx)((\lambda x.x)a) & \\ \swarrow & & \searrow \\ ((\lambda x.x)a)((\lambda x.x)a) & & (\lambda x.xx)a \\ \searrow & & \swarrow \\ & aa & \end{array}$$

<u>3.10</u>. 正規化定理.

リダクションの戦略.

① leftmost (leftmost outermost).

— call by name, normal order.

β基 ($(\lambda x.M)N$ の形のもの) の中で、

λが一番左のもの をβ簡約.

② leftmost innermost (innermost)

— call by value, applicative order

β基を内部に含まない β基 (の最左のもの) をβ簡約.

例. : Ω 無限に簡約. $(\lambda x.y)\Omega \xrightarrow{①} y$. まず中にくりこむ!

②

標準化定理.

もし M が正規形 N をもつならば.

normal order 戦略で N に達する.

(証明略).

では, normal order がよいのか?

normal order : 実引数を使った回数評価.

applicative order : 〃 ならず1回.

cf. call by need (lazy evaluation)

実引数をたかだか1回評価.

言語モデル論 12/19.

1. $\vdash \lambda x.\, x :: nil : int \to list(int)$ を示す.

$$\dfrac{\dfrac{\dfrac{}{x : int \vdash x : int}\,(var) \qquad \dfrac{}{x : int \vdash nil : list(int)}\,(nil)}{x : int \vdash x :: nil : list(int)}\,(int)}{\vdash \lambda x.\, x :: nil : int \to list(int)}\,(abs)$$

注. $\lambda x.\, x :: nil$ は他の型も持てる.

2. $\vdash (let\ f = \lambda x.\, x :: nil\ in\ f\ 0) : list(int)$.

$$\dfrac{(1.で済) \qquad \dfrac{\dfrac{}{\quad \vdash \quad}\,(var) \qquad \dfrac{}{\quad \vdash 0 : int}\,(con)}{f : int \to list(int) \vdash f(0) : list(int)}\,(app)}{\vdash (let\ f = \lambda x.\, x :: nil\ in\ f\ 0) : list(int)}\,(let)$$

type construction

1.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\quad \vdash f : \tau_4 \to \tau_2}\,(var) \qquad \dfrac{}{\quad \vdash (f x) : \tau_4}\,(app)}{f : \alpha,\ x : \beta \vdash f(fx) : \tau_2}}{f : \alpha \vdash \lambda x.\, f(fx) : \tau_2}\,(abs)}{\vdash \lambda f.\lambda x.\, f(fx) : \tau}\,(abs)$$

計算モデル論 1/10.

例題.

問題2. $\overline{\text{two}} = \lambda f.\lambda x.f(fx)$ の主型?

$$\begin{cases} \tau = \alpha \to \beta \to \tau_3 \\ \tau_2 = \beta \to \tau_3 \\ \alpha = \tau_4 \to \tau_3 \\ \alpha = \beta \to \tau_4. \end{cases}$$

$$\to \begin{cases} \tau = (\tau_4 \to \tau_3) \to \beta \to \tau_3 \\ \tau_2 = \beta \to \tau_3 \\ \alpha = \tau_4 \to \tau_3 \\ \tau_4 \to \tau_3 = \beta \to \tau_4 \end{cases}$$

$$\to \begin{cases} \tau = (\tau_4 \to \tau_3) \to \beta \to \tau_3 \\ \tau_2 = \beta \to \tau_3 \\ \alpha = \tau_4 \to \tau_3 \\ \tau_4 = \beta \\ \tau_3 = \tau_4 \end{cases}$$

$$\to \begin{cases} \tau = (\beta \to \tau_3) \to \beta \to \tau_3 \\ \tau_2 = \beta \to \tau_3 \\ \alpha = \beta \to \tau_3 \\ \tau_4 = \beta \\ \tau_3 = \beta \end{cases}$$

$$\therefore \to \quad \tau = (\beta \to \beta) \to (\beta \to \beta)$$

これが two の主型.

例5. $\text{let } f = \lambda x.\, x \text{ in } [(f\ 3)\ (f\ \text{true})]$

$$\begin{cases} \tau = \tau_3 \times \tau_4 \\ \tau_2 = \alpha \to \alpha \\ \forall \alpha.\, \tau_2 > \text{int} \to \tau_3 \\ \forall \alpha.\, \tau_2 > \text{bool} \to \tau_4 \end{cases}$$

$$\to \begin{cases} \tau = \tau_3 \times \tau_4 \\ \tau_2 = \alpha \to \alpha \\ \forall \alpha.\, \alpha \to \alpha > \text{int} \to \tau_3 \\ \forall \alpha.\, \alpha \to \alpha > \text{bool} \to \tau_4 \end{cases}$$

$$\to \begin{cases} \tau = \tau_3 \times \tau_4 \\ \tau_2 = \alpha \to \alpha \\ \tau_5 \to \tau_5 = \text{int} \to \tau_3 \\ \tau_6 \to \tau_6 = \text{bool} \to \tau_4 \end{cases}$$

となるような τ_5, τ_6 がないといけない。

解いて、

$\tau = \text{int} \times \text{bool}$

$\tau_2 = \alpha \to \alpha$

$\tau_3 = \tau_5 = \text{int}$

$\tau_4 = \tau_6 = \text{bool}$.