計算の理論

総合科目F（数理・情報）

後半担当：小林　直樹

理学部情報科学科

大学院情報理工学系研究科コンピュータ科学専攻

後半部分のWebページ：
http://www-kb.is.s.u-tokyo.ac.jp/~koba/class/ComputationTheory/

講義計画

今井担当分

1. 計算とは(チューリング機械、アルゴリズム)
2. チューリング・チャーチの計算仮説
3. 決定不能な問題
4. 計算困難さ(NP完全問題)
5. 難しい問題をなんとかして解く
6. いろいろな計算パラダイム(確率計算など)

小林担当分

7. 計算のモデル1(ラムダ計算)
8. 計算のモデル2(ラムダ計算)
9. 型付きラムダ計算
10. 計算と論理1(カリー・ハワード同型)
11. 計算と論理2(定理証明支援器を使ってみる)
12. 計算と論理3(論理プログラミング)

理論計算機科学A分野
アルゴリズムと計算量

理論計算機科学B分野
計算モデル、プログラム意味論、計算可能性

本日の話題：l計算の表現力

- 種々のデータ（ブール値、組、自然数など）の encoding
- 不動点演算子と再帰

ブール値とその演算の表現

- ブール値の表現

  「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文

  $\text{if } e_1 \text{ then } e_2 \text{ else } e_3 = e_1\ e_2\ e_3$ とすると

  $\text{if } T \text{ then } e_2 \text{ else } e_3 = (\lambda t.\lambda f.t)e_2\ e_3 \to^* e_2$

  $\text{if } F \text{ then } e_2 \text{ else } e_3 = (\lambda t.\lambda f.f)e_2\ e_3 \to^* e_3$

- ブール演算子

  $\text{Not} = \lambda b.\text{if } b \text{ then } F \text{ else } T\ (= b\ F\ T)$

  $\text{And} = \lambda b_1.\lambda b_2.$
  $\quad \text{if } b_1 \text{ then } (\text{if } b_2 \text{ then } T \text{ else } F) \text{ else } F$

組

- 基本データを組み合わせて複雑なデータを表現するのに使用
  - 複素数は、実部と虚部を表す実数の組として表現可能
  - 整数は、自然数と符号の組として表現可能
- 組(M,N)の表現

  $\lambda f.f\ M\ N$

  組の要素にアクセスする関数fを受け取り、
  M,Nに適用する関数
- 組(M,N)からの要素の取り出し
  - $fst(P) = P\ (\lambda x.\lambda y.x)$：組Pの1番目の要素の取り出し
  - $snd(P) = P(\lambda x.\lambda y.y)$：組Pの2番目の要素の取り出し

  $fst(M,N) = (\lambda f.f\ M\ N)(\lambda x.\lambda y.x)$

  $\to (\lambda x.\lambda y.x)\ M\ N \to^* M$

自然数

- ブール値と同様、「**基本構成子**を受け取り、対応する値を返す関数」として表現
- 自然数の基本構成子：

  Z（zero）とS（successor: 次の数を返す関数）
- 自然数 の表現

successor　zero

$0 = \lambda s.\lambda z.\ z$

$1 = \lambda s.\lambda z.\ s\ z$

$2 = \lambda s.\lambda z.\ s\ (s\ z)$

...

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

自然数に関する演算

- 1を足す関数

$\mathrm{Succc} = \lambda n.$ 「n+1」

$= \lambda n.\ \lambda s.\lambda z.\ s^{n+1}\ z$

$= \lambda n.\ \lambda s.\lambda z.\ s\ (s^{n}\ z)$

$= \lambda n.\ \lambda s.\lambda z.\ s\ (n\ s\ z)$

e.g. Succ 1

$= (\lambda n.\ \lambda s.\lambda z.\ s\ (n\ s\ z))\ \lambda s.\lambda z.s\ z$

$\to \lambda s.\lambda z.\ s\ ((\lambda s.\lambda z.s\ z)\ s\ z))$

$\to^{*} \lambda s.\lambda z.\ s\ (s\ z)$

$= 2$

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

自然数に関する演算

- 足し算

$$\text{Plus} = \lambda m.\lambda n.\ 「m+n」$$
$$= \lambda m.\ \lambda n.\ \lambda s.\lambda z.\ s^{m+n}\ z$$
$$= \lambda m.\ \lambda n.\ \lambda s.\lambda z.\ s^{m}\ (s^{n}\ z)$$
$$= \lambda m.\ \lambda n.\ \lambda s.\lambda z.\ s^{m}\ (n\ s\ z)$$
$$= \lambda m.\ \lambda n.\ \lambda s.\lambda z.\ m\ s\ (n\ s\ z)$$

e.g. Plus 2 1

$$= (\lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))(\lambda s.\lambda z.s\ (s\ z))\ (\lambda s.\lambda z.s\ z)$$
$$\to^* \lambda s.\lambda z.\ (\lambda s.\lambda z.s\ (s\ z))\ s\ \underline{((\lambda s.\lambda z.s\ z)\ s\ z)}$$
$$\to^* \lambda s.\lambda z.\ \underline{(\lambda s.\lambda z.s\ (s\ z))\ s\ (s\ z)}$$
$$\to^* \lambda s.\lambda z.\ s\ (s\ (s\ z))$$
$$= 3$$

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

自然数に関する演算

- かけ算

$$\mathrm{Mult} = \lambda m.\ \lambda n.\ 「m \times n」$$

$$= \lambda m.\ \lambda n.\ 0\underbrace{+m+\ldots+m}_{n}$$

$$= \lambda m.\ \lambda n.\ n\ (\mathrm{Plus}\ m)\ 0$$

$$= \lambda m.\ \lambda n.\ n\ (\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ \lambda s.\lambda z.z$$

e.g. Mult 2 3

$= (\lambda m.\ \lambda n.\ n\ (\mathrm{Plus}\ m)\ 0)\ 2\ 3$

$\rightarrow^*\ 3\ (\mathrm{Plus}\ 2)\ 0 = (\lambda s.\lambda z.\ s\ (s\ (s\ z)))\ (\mathrm{Plus}\ 2)\ 0$

$\rightarrow^*\ \mathrm{Plus}\ 2\ (\mathrm{Plus}\ 2\ (\mathrm{Plus}\ 2\ 0))$

$\rightarrow^*\ 6$

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

自然数に関する演算

- べき乗

$\mathrm{Exp} = \lambda m.\ \lambda n.$ 「m^n」

$= \lambda m.\ \lambda n.\ \underbrace{1 \times m \times \ldots \times m}_{n}$

$= \lambda m.\ \lambda n.\ n\ (\mathrm{Mult}\ m)\ 1$

e.g. Exp 2 3

$= (\lambda m.\ \lambda n.\ n\ (\mathrm{Mult}\ m)\ 1)\ 2\ 3$

$\to^*\ 3\ (\mathrm{Mult}\ 2)\ 1 = (\lambda s.\lambda z.\ s\ (s\ (s\ z)))\ (\mathrm{Mult}\ 2)\ 1$

$\to^*\ \mathrm{Mult}\ 2\ (\mathrm{Mult}\ 2\ (\mathrm{Mult}\ 2\ 1))$

$\to^*\ 8$

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

参考：掛け算、べき乗の別の表現

- Mult = λm. λn. λs. n (m s)
- Exp = λm. λn. n m

足し算(λm. λn. λs.λz. m s (n s z))、掛け算、べき乗の内、
べき乗の表現が一番単純！

0判定

- eqzero? = λn. n (λb.F) T

一回でもこれが適用されたらF が返される

e.g. eqzero? 0

= (λn. n (λb.F) T)(λs.λz.z)

$\rightarrow^*$ (λs.λz.z) (λb.F) T

$\rightarrow^*$ T

eqzero? 2

= (λn. n (λb.F) T)(λs.λz.s (s z))

$\rightarrow^*$ (λs.λz.s (s z)) (λb.F) T

$\rightarrow^*$ (λb.F) ((λb.F) T)

$\rightarrow^*$ F

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

自然数の演算：前者関数Pred

$$\mathrm{Pred}(n) = \begin{cases} 0 & \text{if } n=0 \\ n-1 & \text{if } n>0 \end{cases}$$

- 自然数の組に関する関数

  $\mathrm{Next}(x, y) = (x+1, x)$

  1足した値　1つ前の値

  を考えると

  $\mathrm{Next}^n(0,0) = (n, n-1)$

  したがって、

  $\mathrm{Pred} = \lambda n.\ \mathrm{snd}(\mathrm{Next}^n(0,0))$

  $= \lambda n.\ \mathrm{snd}(n\ (\lambda(x,y).(x+1,x))\ (0,0))$

  とすればよい

  引き算 $\mathrm{Minus} = \lambda m.\ \lambda n.\ n\ \mathrm{Pred}\ m$

Schemeの処理系で確認してみよう

- Scheme: 関数型言語の一種
  http://www.schemers.org/
- $\lambda x.M$ は、文字通り (lambda (x) M) と書く
- 算術式は前置記法に基づくので要注意（演算子が必ず前）
  - e.g. 1+2 は (+ 1 2) と書く

本日の話題：λ計算の表現力

- 種々のデータ（ブール値、組、自然数など）の encoding
- 不動点演算子と再帰

無限簡約列を持つ項

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

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

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

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

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

$\to \ldots$

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

fact 3 を計算してみよう

・黒板で

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、

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

  と（再帰を使わないで）表現可能