計算モデル特論

国立情報学研究所

佐藤一郎

E-mail: ichiro@nii.ac.jp

型なしラムダ計算

1. はじめに
2. 関数と型
3. ラムダ記法
4. ラムダ計算
5. 変換例
6. チャーチロッサ性
7. 正規形の求め方
8. ラムダ計算の計算能力

ラムダ計算(Lambda Calculus)

- 1930年代に数学基礎論の研究から生まれた(A. Church)
- 一般に数学で扱われる関数概念に伴う計算的要素を抽出して作られる計算体系
- 関数型プログラミング言語の基礎理論
  - Lisp、Scheme、ML、Haskellなど
- プログラムの意味論、型理論に関係する

関数と型

- 関数:与えられた引数に適用して値を得るための操作
  - f(x): 関数 f を x に適用して得られた値
  - x のとりうる値の領域A(定義域と呼ぶ)
  - f(x) のとりうる領域B(値域と呼ぶ)
  - このような関数の集合は"A→B"と書き、f の型と呼ぶ
- 例: square(x) = x * x
  - x のとりうる領域は自然数Nのとき、square の型は N→N である。これを、square∈N→Nとかく

疑問

- N→(N×N)はどんな関数か？
  - 1つの自然数を与えると関数が得られる関数
    - $f_x(y)=x\cdot y$で、xをある値kに決めれば、$f_k(y)=k\cdot y$で、N→Nの型を持つ関数となる
- (N×N)→(N×N) はどんな関数か？
  - 関数を引数として、関数が得られる関数
    - twice f(x)＝f(f(x)) なる関数 twice を考える
    - f(x)のところにsquareを引数として与えれば、
    - twice square(x)＝square(square(x))
    - として関数を作り出す。

高階関数(higher-order function)

- 関数を引数とする関数や関数を結果とする関数
- 例:

  twice (f (x)) = f(f(x))

関数を引数とする関数

- 関数の関数などを取り扱っていくうえで、関数を値と同様に取り扱えると便利
- 例:関数の関数 twice f(x)＝f(f(x)) を考える
  - f(x)のところにsquareを引数として与えれば、
  - twice square(x)＝square(square(x))＝x･x･x･x
  - 従って、値域はNの型を持つ。
  - f(x)のところに$f_x(y)=x\cdot y$を引数として与えれば、
  - twice $f_x(y)=f_xf_x(y)=(x\cdot y)\cdot y$
  - 従って、値域はN×Nの型を持つ。
  - ....

ラムダ記法の必要性

- 関数として計算を扱うため、余計なものは取り除く
- 例:f(x) = x * x とするとき、f(x)とは
- xを変数とする関数fを表すのか、それとも
- 関数fのxにおける値を表しているのかが不明確
- 関数自身に名前を付けずに一つのモノ(first class object)として扱う

  λx.(x*x)
- ここで λx とはxが(x*x)の引数であることを示す

ラムダ記法の例

- 例：f(x) = x^2+2y+1のラムダ記法
  - λx.(x^2+2y+1)　　関数(x^2+2y+1)の引数はxであり、yは固定値と扱う
- c.f.
  - λy.(x^2+2y+1)　　関数(x^2+2y+1)の引数はyであり、xは固定値と扱う
  - λx.(λy.(x^2+2y+1))　　関数(x^2+2y+1)の引数はxとyであること

ラムダ抽象(Lambda Abstraction)

- 式Mの中の固定値を表す名前を変数にすること

  λx.M

- Mはxを変数とする関数となる
- ラムダ抽象の逆操作
- ラムダ適用、
- 部分計算
- 定数畳み込み

ラムダ適用(Lambda Application)

- 関数M中の変数 x に値（または関数） d を代入すること

  ((λx.M)d)

- 例：
  - ((λx.(x^2+2y+1))3)　→　　3^2+2y+1
  - ((λy.(x^2+2y+1))4)　→　　x^2+2･4+1
  - ((λx.(λy.(x2+2y+1))4)3) →3^2+2･4+1

関数の自己適用

- 関数twiceのラムダ記法
  - twice＝λf.（λx. f(f x)）
  - 関数twiceに関数gを引数として適用すると、
    - twice g n＝(λf.（λx. f(f x))g)n
      ＝(λx. g(g x))n ＝ g(g n)
  - gをtwiceに置き換えてみる
    - twice twice g n ＝((twice twice)g)n
      ＝(twice(twice g))n＝(twice g)((twice g)n)
      ＝(g(g((twice g)n)))＝g(g(g(g n)))

ラムダ式

- BNF文法による定義

$$M ::= x \mid \underbrace{(\lambda x.M)}_{\text{ラムダ抽象}} \mid \underbrace{(M_1\ M_2)}_{\text{ラムダ適用}}$$

- ラムダ計算とは規則に従って、ラムダ式を順次変形していくこと

ラムダ式

- ラムダ式の定義
  - (1)変数x, y, z. . . , 定数1, 2, 3, . . . はラムダ式
  - (2)Mがラムダ式、xが変数なら、λx. Mもラムダ式
    (ラムダ抽象)
  - (3)M, Nがラムダ式なら、MNもラムダ式
    (関数適用)

- 表記
  - $\lambda x_1 x_2 \cdots x_n.\ M = \lambda x_1.\ (\lambda x_2.\ (\cdots(\lambda x_n.\ M)\cdots))$
  - $M_1 M_2 M_3 \cdots M_n = (\cdots((M_1 M_2) M_3)\cdots) M_n$

ラムダ式の例

- x
- (λx.x)
- (λx.y)
- (λx. (λy.x))
- ((λx.(x x)) y)
- ((λx.(x x)) (λy.y))

ラムダ式の省略形

- 省略の規則:
- 一番外側の括弧は外してよい
- $(\lambda x_1.(\lambda x_2 \ldots(\lambda x_n.M)\ldots))$は$\lambda x_1 x_2 \ldots x_n.M$と書いてよい
- $(\ldots(M_1\ M_2)\ldots M_n)$は$M_1\ M_2\ \ldots\ M_n$と書いてよい

例題:

$(\lambda x.(\lambda y.((xy)(zu))))$

$= \lambda x.(\lambda y.((xy)(zu)))$

$= \lambda x \lambda y.((xy)(zu))$

$= \lambda x \lambda y.xy(zu)$

自由変数

- ラムダ式Mに含まれる自由変数の集合FV(M)

$$FV(x) = \{x\}$$
$$FV(M_1\ M_2) = FV(M_1) \cup FV(M_2)$$
$$FV(\lambda x.M) = FV(M) - \{x\}$$

- 自由変数でない変数を束縛変数

変数

- 束縛変数
- ラムダ式のxを変数としてラムダ抽象
- 自由変数
  - 式に含まれる変数と抽象化の対象が結びついているかどうか

    x(λxy. xyz)xy
    ①　②③ ④⑤⑥ ⑦⑧

    このとき、自由変数は①、⑥、⑦、⑧、束縛変数は、②、③、④、⑤

- ラムダ式M_1, M_2, ・・・M_nで、それらの束縛変数と自由変数が重ならないように置き換えができる。
- 重なりがない状態＝「変数条件を満たす」という

変換規則

- α-規則（束縛変数の名前を置換）

$$(\lambda x.M)=(\lambda y.[y/x]M)$$

- ただし、yがMの自由変数ではないとする

- β-規則（ラムダ式における計算）

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

- ここで、[b/a]MとはM中の自由変数aをbで置き換える

- β変換によるラムダ式の書き換えをリダクションという。
- リダクションが可能な部分をリデックスと呼ぶ。
- リデックスが含まれていないとき、そのラムダ式は正規形であるという

α変換の例

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

$((\lambda x.x)\ (\lambda x.xy)) = ((\lambda y.y)\ (\lambda z.zw))$

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

リダクションの練習問題

(1)(λxy. y)３２
(2)(λxy. xy)(λw. w･w)９
(3)(λxy. x)(λx. xx)(λz. z)
(4)(λxy. y)(λx. xx)(λz. z)
(5)(λx. x(λxy. x))(λx. x)
(6)(λx. x(λxy. x))(λx. x(λxy. y))(λx. x)

リダクションの練習問題(解答)

(1)(λxy. y)３２ → (λy. y)２ → ２

(2)(λxy. xy)(λw. w･w)９
→(λy. (λw. w･w)y)９→(λy. y･y)９→９･９

(3)(λxy. x)(λx. xx)(λz. z)
→(λy. (λx. xx))(λz. z)→λx. xx

(4)(λxy. y)(λx. xx)(λz. z)
→(λy. y)(λz. z)→λz. z

リダクションの練習問題(解答)

- (5)(λx. x(λxy. x))(λx. x)
- →(λx. x)(λxy. x)→(λxy. x)

- (6)(λx. x(λxy. x))(λx. x(λxy. y))(λx. x)
- →(λx. x(λxy. y))(λxy. x)(λx. x)
- →(λxy. x)(λxy. y)(λx. x)
- →(λy'. (λxy. y))(λx. x)
- →(λxy. y)

変換(リダクション)の例

- (1)自由変数と束縛変数が衝突する場合は、書き換えておく
  - (λxy. x)yz → (λy. y)z → z (誤り)
  - (λxy. x)yz → (λxy'. x)yz → (λy'. y)z→y
- (2)リダクションを行うと複雑になってしまう例
  - (λx. xxx)(λx. xxx)
  - →(λx. xxx)(λx. xxx)(λx. xxx)
  - →(λx. xxx)(λx. xxx)(λx. xxx)(λx. xxx)
  - →・・・・

変換（リダクション）の例（続き）

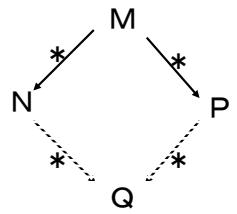
- （3）自分に戻ってしまうリダクション
  - （λx. xx）（λx. xx）→（λx. xx）（λx. xx）
- （4）異なる部分から始めて同じ結果が出るリダクション
  - I≡λx. xとする
  - I（I x）は二つのリデックスI（I x）とI xを持つ
  - I（I x）→ I x → x
  - I（I x）→ I x → x

正規形の求め方

- 正規形を計算する戦略
  - 2つのリデックスがあるとき、どちらを選ぶか？
    - M≡（λx. y）（（λx. xx）（λx. xx））　①　②
    - ①では、M→M→・・・の無限のリダクションが続く
    - ②では、M→yとなり、1回で完了
- リダクション戦略
  - （1）ラムダ式がM正規形を持つならば、最も左側のリデックスを常にリダクションすることで、必ず正規形が得られる。
  - （2）最も左側のリデックスとは、最も外側のリデックスのうちで、最も左側のものであること

チャーチ・ロッサ性

- ラムダ式にリデックスが複数あるとき、そのどれに注目するかにより、何通りかのリダクションの可能性がある。
- 場合によっては、正規形にならないリダクションもある。
- 複数のリダクション列ができるとき、その結果得られる正規形は途中のリダクションの道筋によらず一意に決まる。



M→*N、M→*Pのとき、
（0回以上のリダクションで
MからNに到達する意味）
（MからP）
適当なリダクションで、
Qに合流できる。

チャーチ・ロッサ性の利点

- リダクションの順序に気を使う必要がない
- どんな方法でリダクションを行っても、得られた結果（正規形）が唯一であることが保証される
- （通常の並列計算や、非決定的な計算では、計算の順序を保つため、同期が必要となる）

ラムダ計算の計算能力

- ラムダ計算モデルによるプログラム
- 各自然数kを正規形のラムダ式「k」で表す。 g:$N^n \to N$に対して、

$$g(k_1, k_2, \cdots, k_n) = k \Leftrightarrow G「k_1」「k_2」\cdots「k_n」\to「k」$$

- を満たすラムダ式Gを、関数gを計算するためのプログラムとみなす。
- 「k_1」, 「k_2」, ・・・, 「k_n」はこのプログラムの入力とみなす。

- このプログラムGを入力「k_1」, 「k_2」, ・・・, 「k_n」に対して β変換を次々と行うことを、計算過程としてとらえる。
- 変換が終結して「k」が得られたとき、プログラムの計算結果がkであると考える。変換が終結しない場合、プログラムの計算結果は未定義となる。

コード化:論理値

- 論理値の真(true)と偽(false)は次のようなラムダ式に対応
  「true」≡ λxy. x (Tともかく)
  「false」≡ λxy. y (Fともかく)

- 条件判定式に対応するラムダ式(AとBはプログラム分に相当)
  「cond」≡ λb λA. λB. bAB

- 例:任意のAとBに対して
  Cond true A B →・・・ → A
  Cond false A B →・・・ → B

コード化:自然数

- 自然数nに対応するλ式
- 「0」と「次の自然数」という概念からコード化
  「0」≡ λf. λz. z
  「1」≡ λf. λz. f z
  「2」≡ λf. λz. f (f z)
  ・・・
  「N」≡ λf. λz. $f^n z$

- 次の自然数を求める関数のコード化
  - Succ≡ λn. λf. λz. f(n f z)

コード化:自然数演算

- 自然数演算に対応するλ式
- 足し算のコード化
  - Add≡ λm. λn. m Succ n
- かけ算のコード化
  - Mul≡ λm. λn. m (Add n) 「0」
- ゼロ判定関数のコード化
  - IsZero≡ λn. n (λn. 「false」)「true」

不動点オペレータ

- 例（Yコンビネータ）
  - Y≡λf.（λx. f(x x)）(λx. f(x x)）
  - Yを任意のラムダ式Fに適用すると
    - YF → (λx. F(x x))(λx. F(x x))
    - → F((λx. F(x x))(λx. F(x x)))
    - ← F(YF)
  - β変換を等式とみなすと、
    - F(YF)＝YF ・・・・ラムダ式Fの不動点はYFとなる
    - （関数fの不動点とは、f(x)＝xとなるxのこと）

計算モデル特論

国立情報学研究所

佐藤一郎

E-mail: ichiro@nii.ac.jp

π計算

- CCSに通信名（通信ポート名）の受け渡し機構を拡張 → π計算
  - Engberg,Nielsen[1986年]
  - 通信引数に通信ポート名を導入
  - Thomsen[1989年]
  - プロセス受け渡し機構付きCCS（高階CCS）
  - Milner,Parrow,Walker[1989年]
  - π計算の提案（構造的操作意味論による定義）
  - Milner[1990年]
  - 構造合同により簡約系としてπ計算を再定義
  - Milner[1991年]
  - 多引数（polyadic)π計算

π計算の構文

- ただし、π計算には構文構成子が異なる変形が存在する

P	:=	0	（停止したプロセス）
	|	$\overline{x}y.P$	（出力ポートxから引数yを送信）
	|	$x(y).P$	（入力ポートxから値または通信ポート名をyで受信）
	|	$P+Q$	（PまたはQとして振る舞う）
	|	$P\|Q$	（PとQが並行に動作できる）
	|	$(x)P$	（通信ポートxをPのスコープ内に制限する）
	|	$[x=y]P$	（xとyが一致したときのみPとして振る舞う）
	|	A	（A=Pとなるとき、AをPで置換）

事象の名前

- π計算には遷移ラベル：

$$\alpha ::= \tau \quad \text{内部計算（外部から観測されない）}$$
$$\mid \quad \bar{x}y \quad \text{束縛されない名前 y を出力ポート x から送信}$$
$$\mid \quad \bar{x}(y) \quad \text{束縛された名前 y を出力ポート x から送信}$$
$$\mid \quad x(y) \quad \text{入力ポート x から受け取った名前を束縛名 y と置換}$$

α	$fn(\alpha)$	$bn(\alpha)$	$n(\alpha)$
τ	$\varnothing$	$\varnothing$	$\varnothing$
$\bar{x}y$	$\{x,y\}$	$\varnothing$	$\{x,y\}$
$\bar{x}(y)$	$\{x\}$	$\{y\}$	$\{x,y\}$
$x(y)$	$\{x\}$	$\{y\}$	$\{x,y\}$

π計算の遷移

- π計算の状態遷移

$$\bar{x}y.P \xrightarrow{\bar{x}y} P \qquad x(z).\bar{z}5Q \xrightarrow{x(w)} \bar{w}5.Q\{w/z\}$$

$w \notin fn(Q) \vee w = y$

$$\bar{x}y.P \mid x(z).\bar{z}5.Q \xrightarrow{\tau} P \mid \bar{y}5.Q\{y/z\}$$

$fn(P)$ とは P に含まれる未束縛な名前の集合

π計算の操作意味論（構造操作意味論）

- 構造的操作意味論による定義 $\alpha \in \{\bar{x}y, x(y), \tau\}$

$$\bar{x}y.P \xrightarrow{\bar{x}y} P \qquad x(y).P \xrightarrow{x(w)} P\{w/y\}\left(w = y \cup w \notin fn(P)\right) \qquad \tau.P \xrightarrow{\tau} P$$

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{[x=x]P \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\left(bn(\alpha) \cap fn(Q) = \varnothing\right) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P'|Q}\left(bn(\alpha) \cap fn(P) = \varnothing\right)$$

$$\frac{P \xrightarrow{\bar{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P|Q \xrightarrow{\tau} (w)(P'|Q')} \qquad \frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$$

$$\frac{P \xrightarrow{\bar{x}y} P'}{(y)P \xrightarrow{\bar{x}(w)} P'\{w/y\}}\left(\begin{matrix} y \neq x \\ w \notin fn(P') \vee w = y \end{matrix}\right) \qquad \frac{P \xrightarrow{\alpha} P'}{(x)P \xrightarrow{\alpha} (x)P'} x \notin n(\alpha)$$

$$\frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}(A =_{def} P)$$

+と｜は対称

π計算の操作意味論（通信 1/2）

- 構造的操作意味論による定義 $\alpha \in \{\bar{x}y, x(y), \tau\}$

OUT: $\bar{x}y.P \xrightarrow{\bar{x}y} P$

IN: $x(y).P \xrightarrow{x(w)} P\{w/y\}\left(w = y \cup w \notin fn(P)\right)$

yまたはP内の非束縛名に変更可能

COM: $$\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$$

PAR: $$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\left(bn(\alpha) \cap fn(Q) = \varnothing\right)$$

束縛名を受け取るときには、他の並行プロセスの非束縛名と重ならないこと

遷移例

$$\bar{x}y.0 \mid x(z).\bar{z}5.0 \xrightarrow{\tau} 0 \mid \bar{y}5.0$$
$$\xrightarrow{\bar{y}5} 0 \mid 0$$

π計算の操作意味論(通信 2/2)

$$\bar{x}y.P \xrightarrow{\bar{x}y} P \qquad x(y).P \xrightarrow{x(w)} P\{w/y\}\,(w = y \cup w \notin fn(P))$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\,(bn(\alpha) \cap fn(Q) = \varnothing) \qquad \frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(w)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/w\}}$$

遷移例 $\bar{x}y.P|(x(z).Q|R)$

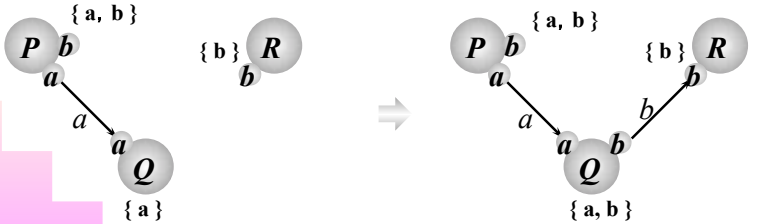
$$\frac{\bar{x}y.P \xrightarrow{\bar{x}y} P \qquad \dfrac{x(z).Q \xrightarrow{x(w)} Q\{w/z\}}{x(z).Q|R \xrightarrow{x(w)} Q\{w/z\}|R}\,\begin{pmatrix} w=z \cup w \notin fn(P) \\ bn(x(w)) \cap fn(Q)=\varnothing \end{pmatrix}}{\bar{x}y.P|(x(z).Q|R) \xrightarrow{\tau} P|(Q\{w/z\}|R)\{y/w\}}$$

$$\bar{x}y.P|(x(z).Q|R) \xrightarrow{\tau} P|(Q\{y/z\}|R) \quad ○$$
$$\xrightarrow{t} P|(Q|R)\{y/z\} \quad ×$$

動作式例(1)

- π計算の動作式の基本例:

$$\bar{a}b.P|a(x).\bar{x}5.Q|by.R \xrightarrow{\tau} P|(\bar{x}5.Q)\{b/x\}|by.R$$
$$\xrightarrow{\tau} P|(\bar{x}y.Q)\{b/x\}|by.R$$
$$(i.e.\ \ P|\bar{b}5.Q\{b/x\}|by.R)$$
$$\xrightarrow{\tau} P|Q\{b/x\}|R\{5/y\}$$



π計算の遷移

- π計算の状態遷移 $(x)P$ とは P 内で名前 x を宣言すること

RES: $\dfrac{P \xrightarrow{\alpha} P'}{(x)P \xrightarrow{\alpha} (x)P'}\,(x \notin n(\alpha))$ 通信名がxのときは通信制限
引数がxのときはOPENとCLOSEによる

OPEN: $\dfrac{P \xrightarrow{\bar{x}y} P'}{(y)P \xrightarrow{\bar{x}(w)} P'\{w/y\}}\begin{pmatrix} y \neq x \\ w \notin fn(P') \vee w = y \end{pmatrix}$

束縛名をスコープ外に出す

CLOSE: $\dfrac{P \xrightarrow{\bar{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P|Q \xrightarrow{\tau} (w)(P'|Q')}$

束縛名を受け取り、スコープを広げる

π計算の遷移

- π計算の遷移ラベル

$P \xrightarrow{\tau} P'$ 環境と通信を行うことなく P から P' に状態遷移する
τP または P 内通信によって生じる

$P \xrightarrow{x(y)} P'$ リンク x に非束縛名 y を受信し P から P' に状態遷移する
$x(y)$ によって生じる

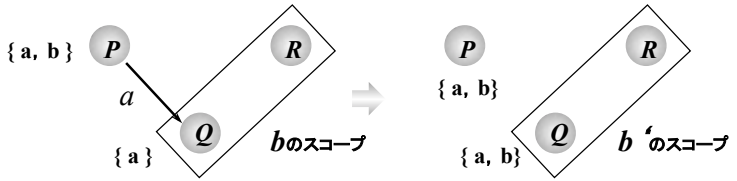
$P \xrightarrow{\bar{x}y} P'$ リンク x に非束縛名 y を送信し P から P' に状態遷移する
$\bar{x}(y)$ によって生じる

$P \xrightarrow{\bar{x}(y)} P'$ リンク x に束縛名 y を送信し P から P' に状態遷移する
$(y)\bar{x}y$ によって生じる

動作式例(2)

- π計算の動作式の基本例:

$$\bar{a}b.P|(b)(a(x).Q|R) \xrightarrow{\tau} P|(b')(Q\{b'/b\}\{b/x\}|R\{b'/b\})$$

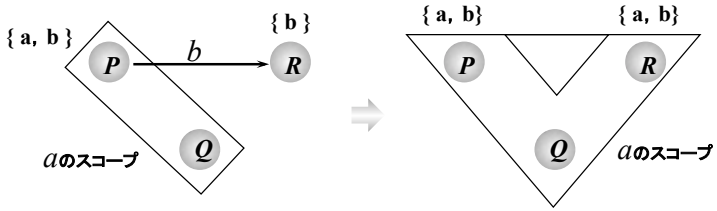


π計算ではα変換が可能: $(x)(P|Q) = (x')(P\{x'/x\}|Q\{x'/x\})$

動作式例(3)

- π計算の動作式の基本例:

$$(a)(\bar{b}a.P|Q)|b(x).R \xrightarrow{\tau} (a)(P|Q|R\{a/x\})$$



π計算の操作意味論(構造操作意味論)

-

$$\text{SUM: } \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$$

$$\text{MATCH: } \frac{P \xrightarrow{\alpha} P'}{[x=x]P \xrightarrow{\alpha} P'}$$

$$\text{IDE: } \frac{P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'} (A(\tilde{y}) =_{def} P) \qquad \tilde{y} = y_1 \cdots y_n$$

π計算の操作意味論(構造合同性)

- 構文的に一致する動作式集合を操作意味論に導入(λ計算のα変換に相当)

$$P|Q \equiv Q|P \quad P|(Q|R) \equiv (P|Q)|R \quad P|0 \equiv P$$

$$P+Q \equiv Q+P \quad P+(Q+R) \equiv (P+Q)+R \quad P+P \equiv P \quad P+0 \equiv P$$

$$(x)(y)P \equiv (y)(x)P \quad (x)0 \equiv 0 \quad (x)(P|Q) \equiv P|(x)Q \text{ if } x \notin fn(P)$$

$$[x=x]P \equiv P$$

$$A \equiv P(\text{where } A =_{def} P)$$

$$\text{STRUCT: } \frac{P \equiv Q \quad P \xrightarrow{\alpha} P' \quad P' \equiv Q'}{Q \xrightarrow{\alpha} Q'}$$

π計算の操作意味論（構造合同性）

- 構文的に一致する動作式集合を操作意味論に導入（λ計算のα変換に相当）

$$\bar{x}y.P \xrightarrow{\bar{x}y} P \qquad x(y).P \xrightarrow{x(w)} P\{w/y\}\left(w = y \cup w \notin fn(P)\right)$$

$$(\cdots + x(y).P) \mid (\cdots + \bar{x}z.Q) \xrightarrow{\tau} P\{z/y\} \mid Q$$

$$\frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\left(bn(\alpha) \cap fn(P) = \varnothing\right) \qquad \frac{P \xrightarrow{\alpha} P'}{(x)P \xrightarrow{\alpha} (x)P'}\, x \notin n(\alpha)$$

$$\frac{P \xrightarrow{\bar{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P \mid Q \xrightarrow{\tau} (w)(P' \mid Q')} \qquad \frac{P \xrightarrow{\bar{x}y} P'}{(y)P \xrightarrow{\bar{x}(w)} P'\{w/y\}}\left(\begin{array}{c} y \neq x \\ w \notin fn(P') \vee w = y \end{array}\right)$$

$$\frac{P \equiv Q \quad P \xrightarrow{\alpha} P' \quad P' \equiv Q'}{Q \xrightarrow{\alpha} Q'}$$

π計算の構文（追加）

- 複製（Replication）

$!P$ $\qquad !P \equiv P \mid P !$

制限（Restriction）

$(\nu x)P$ $\qquad (\nu x)P$ is $(x)P$

多引数π計算（Polyadic π Calculus）

$$\begin{array}{lll} P & ::= & \bar{x}.[y_1 \cdots y_n]P \qquad \bar{x}y_1 \cdots y_n.P \\ & \mid & x(y_1 \cdots y_n).P \\ & \mid & (\text{同}) \end{array}$$

$$\bar{x}.[y_1 \cdots y_n]P \mid x(z_1 \cdots z_n).Q \xrightarrow{\tau} P \mid Q\{y_1/z_1, \ldots, y_n/z_n\}$$

プロセス等価性

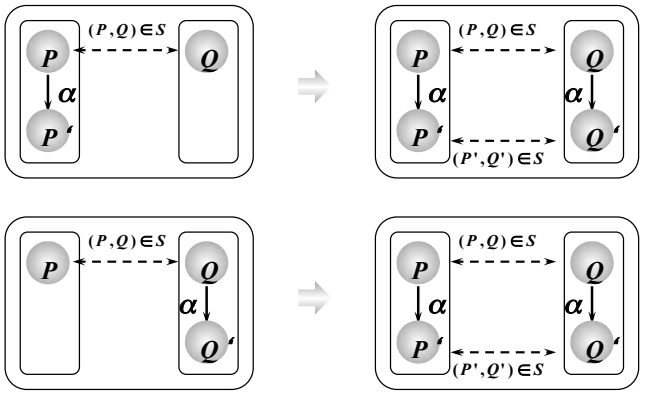
- プロセス間の等価性
- 例：仕様を表したプロセス vs 実装を表したプロセス
- → 実装は仕様を満足している

- CCSなどのプロセス等価性をπ計算に導入：
- 双模倣性、試験等価、失敗等価

- ただし、π計算の等価性では名前を受け渡しを扱う必要がある

双模倣性（Bisimulation）

$P \overset{(P,Q)\in S}{\cdots\cdots} Q$ とは



Early Bisimulation

早期双模倣関係 (P,Q) ∈ S とは、任意の通信名a、引数xに対して以下の3条件が成立すること

(1) $P \xrightarrow{\bar{a}x} P' \Rightarrow \exists Q' : Q \xrightarrow{\bar{a}x} Q' \quad and \quad (P', Q') \in S$

(2) $P \xrightarrow{a(x)} P' \quad and \quad x \notin FN(P, Q) \quad then$
$\exists Q' : Q \xrightarrow{a(x)} Q' \quad and \quad \forall y : (P'\{y/x\}, Q'\{y/x\}) \in S$

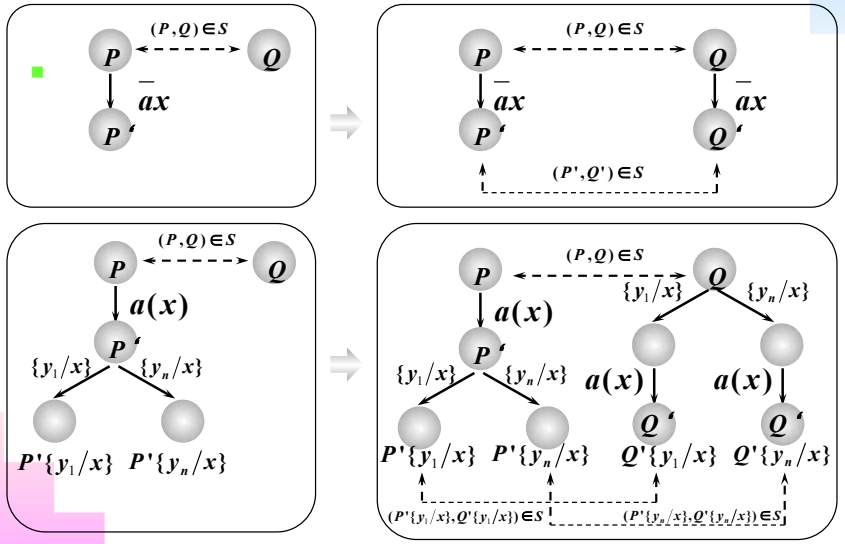
(3) S は 対称

となる早期双模倣関係 S が存在するとき、 P ~E Q とし、PとQは早期双模倣であるという

Ichiro Satoh

Early Bisimulation

$P \overset{(P,Q)\in S}{\cdots\cdots} Q$ とは



Ichiro Satoh

Late Bisimulation

遅延双模倣関係 (P,Q) ∈ S とは、任意の通信名a、引数xに対して以下の3条件が成立すること

(1) $P \xrightarrow{\bar{a}x} P' \Rightarrow \exists Q' : Q \xrightarrow{\bar{a}x} Q' \quad and \quad (P', Q') \in S$

(2) $P \xrightarrow{a(x)} P' \quad and \quad x \notin FN(P, Q) \quad then$
$\forall y, \exists Q' : Q \xrightarrow{a(x)} Q' \quad and \quad (P'\{y/x\}, Q'\{y/x\}) \in S$

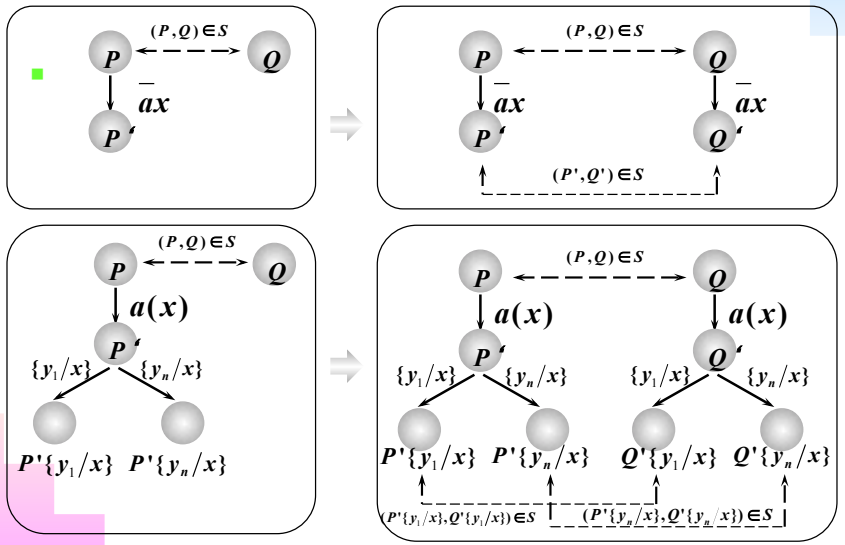
(3) S は 対称

となる遅延双模倣関係 S が存在するとき、 P ~L Q とし、PとQは遅延双模倣であるという

Ichiro Satoh

Late Bisimulation

$P \overset{(P,Q)\in S}{\cdots\cdots} Q$ とは



Ichiro Satoh

最近の研究

- 型理論との融合
  - 並行計算特有の型理論は存在するのか
- 高階プロセス計算
  - プロセスそのものを通信引数として受け渡し
- プログラミング言語の意味論
  - 関数型プログラミング言語、オブジェクト指向言語の意味論を定式化
- プログラミング言語としての実現
  - π計算に基礎におくプログラミング言語処理系の実装
- 非同期通信機構の導入
  - 同期通信より非同期通信にもとづく方が理論的洗練化が可能？

π計算の問題点

- 名前渡し機構は本当に必要なのか？
- 例:通信プロトコルの記述言語:
  - プロセス代数はOSIデータリンク層のプロトコルが対象
    - データリンク層の通信リンクは固定的
    - 名前渡し機能は必要はない
- 例:プログラミング言語として利用(例:PICT等):
  - 意味論上の制約(ガード付き出力通信、事象の原子性など)の実装は困難
  - 他の並列・分散プログラミング言語でも十分
- 検証システムは構築可能なのか(π計算版のCWB等)？
  - CCSでも検証システムの構築は困難
- 体系自体が複雑すぎる？

π計算の問題点

- π計算は相互作用系を表す最小の体系なのか？

$P|Q \quad !P$ Parallelism

$(\nu x)P$ Naming, Restriction, Abstraction

$\bar{x}y.P \quad x(y).P$ Sending, Receiving

しかし、Binding 概念と Guarding 概念が混在

$$\bar{x}y|x(z).P \longrightarrow P\{y/z\}$$

Binding 概念と Guarding 概念の分離 → Action Structure [Milner93]

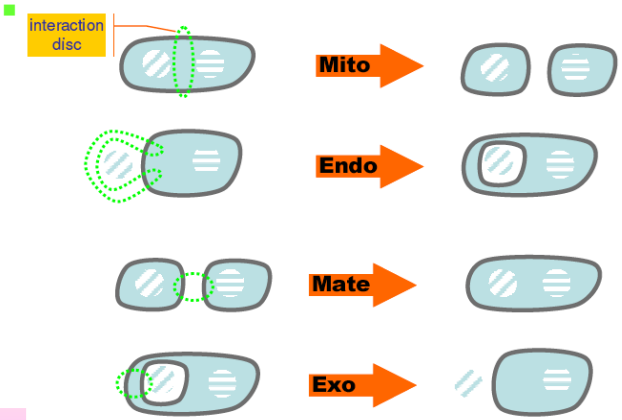
計算モデル特論

国立情報学研究所

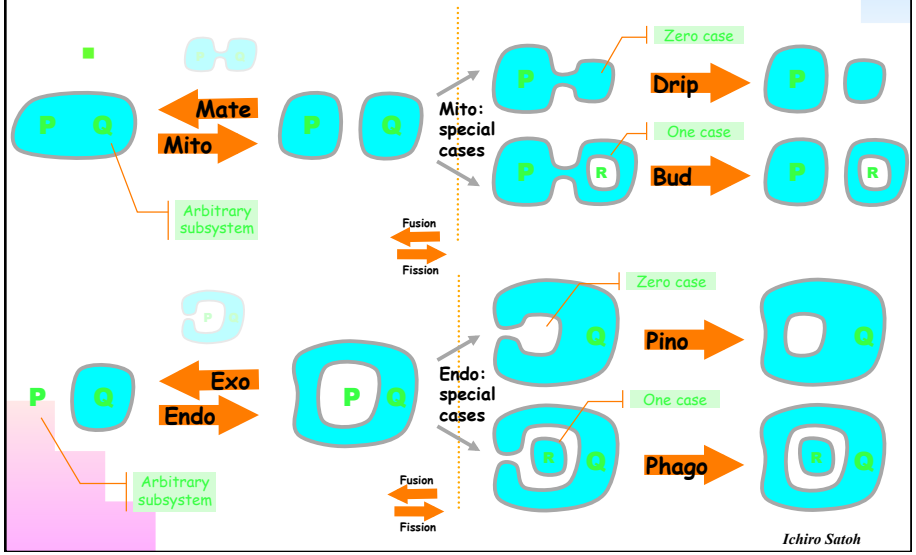
佐藤一郎

E-mail: ichiro@nii.ac.jp

細胞膜と動作



細胞膜と動作



Membrane Calculus

- 構文

X ::=		membrane system
	◇	empty system
	X∘X	composition of subsystems
	⦇X⦈	subsystem inside membrane

Membrane Calculus

- 操作意味論

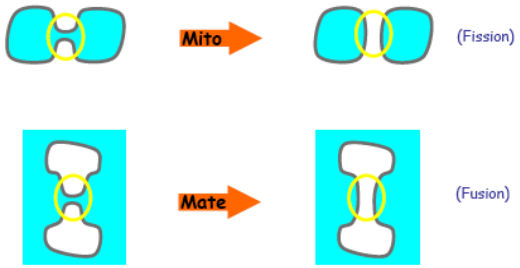
$$X_1 \circ X_2 \equiv X_2 \circ X_1$$
$$X_1 \circ (X_2 \circ X_3) \equiv (X_1 \circ X_2) \circ X_3$$
$$X \circ \diamond \equiv X$$

$$X_1 \rightarrow X_2 \Rightarrow X_1 \circ X \rightarrow X_2 \circ X$$
$$X_1 \rightarrow X_2 \Rightarrow ⦇X_1⦈ \rightarrow ⦇X_2⦈$$
$$X_1 \equiv X'_1 \rightarrow X'_2 \equiv X_2 \Rightarrow X_1 \rightarrow X_2$$

細胞分裂と結合

- 分裂(Mite)と結合(Mate)

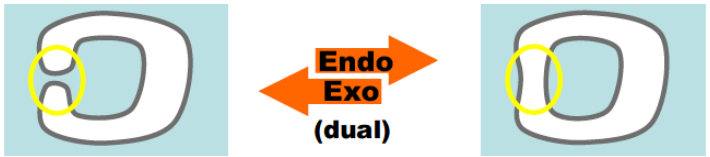
(X)∘(X') ⇆ (X∘X')　　Mito/Mate
X ⇆ ((X))　　Peel/Pad



Endo/Exo

- (逆)内包化

◇ ⇆ (◇)　　Froth/Fizz
X∘(Y) ⇆ ((X)∘Y)　　Endo/Exo



例題

- 基本動作の表現

(X)∘(X') ⇆ (X∘X')　　Mito/Mate
X ⇆ ((X))　　Peel/Pad

- 証明(表現可能性)

(X)∘(X') ⇆ ((X))∘X') ≡ ((◇∘(X))∘X')
⇆ ((◇)∘X∘X') ⇆ ◇∘(X∘X') ≡ (X∘X')

X ≡ X∘◇ ⇆ X∘(◇) ⇆ ((X)∘◇) ≡ ((X))

表現性

The reactions Mito and Mate can be represented by Endo and Exo.