再帰プログラムの幾何

Geometry of Recursive Programs

長谷川 真人

京都大学数理解析研究所

2006 年 9 月 15 日

I. 序論：プログラミング言語の意味論の研究

目的

複雑なプログラムの背後にある数学的な構造をつきとめ、
分析することによって、プログラムに関して成り立つ
本質的な原理を導く

応用

- プログラムを見通し良く**設計**
- プログラムが仕様どおりに動くことを**検証**
- プログラムをより効率の良いもの・わかりやすいものに**改良**

方針

- 適切な数学モデルを用いてプログラムの構造を**抽象化**
- 基礎理論研究の立場：本質、普遍性、簡潔性を重視

プログラムにどう立ち向かうか

プログラム — コンピュータが行なう仕事（コンピュテーション）を直接的・具体的に記述する最強・最良の手段の一つ

プログラムの**制御構造**や**データ構造**は
コンピュテーションに明確な「かたち」を与える

けれども、コンピュテーションそのものについて考えるためには、
プログラムに直接立ち向かうことは常にベストの方法とはいえない

- プログラムはたいてい複雑で、しばしば巨大である
- プログラムは、たいてい冗長である
- プログラムは、本質的に理解困難である

プログラミング言語の意味論

プログラムのあらわすコンピュテーションの本質的な意味を
システマティックに解読する理論

プログラムを直接読んだり実行したりするかわりに、
適切な抽象化によって得られた、より扱いやすい情報を用いて、
そのふるまいを理解する

適切な抽象化は、問題の本質を浮き彫りにする

抽象化の利点

「幾何学は与えられた変換群に属する任意の変換で不変な
図形の性質を研究する学問である」
（クライン）

「物理学の学習はありのままに世界をみない手法を
身につけるための修行である」
（佐藤文隆）

抽象化は、「○○とはなにか」という問いではなく、
「○○はどのような性質をもつか、どのように理解できるか」
という問いに答えるために有効な考え方である

この講演では

プログラミング言語の主要な制御構造のひとつである再帰（recursion）に焦点をあて、その意味論の最近の（講演者自身が関わってきた）展開について考える

I 序論

II 再帰プログラムの不動点意味論（背景）

III 再帰プログラムの幾何（本題）

IV まとめ

II. 再帰プログラムの不動点意味論

再帰プログラムの不動点意味論

再帰プログラム、たとえば階乗を計算するプログラム

$$\mathsf{fact}(x) \equiv \text{if } x = 0 \text{ then } 1 \text{ else } x \times \mathsf{fact}(x-1)$$

は、以下の汎関数 F の**不動点**として理解することができる：

$$F(f)(x) \equiv \text{if } x = 0 \text{ then } 1 \text{ else } x \times f(x-1)$$

すなわち $\mathsf{fact} = F(\mathsf{fact})$ となる。

不動点意味論

再帰プログラムを、適切な数学構造の上の写像の**不動点**として抽象化して解釈（**表示的意味論・領域理論**、１９７０ごろ〜）

$$
\begin{array}{lcl}
 & & \mathsf{fact}(4) \\
(\mathsf{fact} = F(\mathsf{fact})) & = & F(\mathsf{fact})(4) \\
(F \text{ の定義}) & = & \text{if } 4 = 0 \text{ then } 1 \text{ else } 4 \times \mathsf{fact}(4-1) \\
(4 \neq 0) & = & 4 \times \mathsf{fact}(3) \\
(\mathsf{fact} = F(\mathsf{fact})) & = & 4 \times F(\mathsf{fact})(3) \\
(F \text{ の定義}) & = & 4 \times (\text{if } 3 = 0 \text{ then } 1 \text{ else } 3 \times \mathsf{fact}(3-1)) \\
(3 \neq 0) & = & 4 \times (3 \times \mathsf{fact}(2)) \\
 & \cdots & \\
 & = & 4 \times (3 \times (2 \times (1 \times 1))) \\
 & = & 24
\end{array}
$$

領域理論

領域理論（domain theory）では

- **データの集まりを**
  適当な条件（完備性）を満たす順序集合（データ領域）として
- **プログラムを**
  領域のあいだの適当な条件（連続性）を満たす関数として

抽象化して表現する。

イメージ：

$$\text{計算可能な関数全体} \subset \text{連続な関数全体} \subset \text{関数全体}$$

（離散的な複雑な現象を、連続的な世界で、筋良く近似的にとらえる）

（連続・線型・可換なもののほうが扱いやすい）

完備半順序集合

定義　半順序集合 $(X, \sqsubseteq)$ が最小の要素を持ち、X の可算単調列

$$x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq \ldots$$

が上限を持つとき、$(X, \sqsubseteq)$ は完備半順序集合
（complete partial order, CPO**）であるという。**

CPO $(X, \sqsubseteq)$ **の最小元を** $\bot_X$ **または** $\bot$ **であらわす。**

直観的には、$\bot$ は、「未定義」「止まらない計算」をあらわしている。

（「未定義」を平気で扱うのは計算機科学の特質である）

完備半順序集合の例

自然数上の部分関数全体の集合 $[\mathbf{N} \rightharpoonup \mathbf{N}]$ に、以下のような順序 $\sqsubseteq$ を定める：

$$f \sqsubseteq g \iff \text{任意の } n \in \mathbf{N} \text{ について } f(n) \text{ が定義されているならば } g(n) \text{ も定義されていてしかも } f(n) = g(n)$$

この順序に関して $[\mathbf{N} \rightharpoonup \mathbf{N}]$ は完備半順序集合になる。最小元は、任意の $n \in \mathbf{N}$ について対応する値が未定義であるような部分関数である。また、$f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ について、

$$(\bigsqcup f_i)(n) = \begin{cases} f_k(n) & \text{ある } k \text{ について } f_k(n) \text{ が定義されているとき} \\ \text{未定義} & \text{その他の場合} \end{cases}$$

連続関数

CPO $(X, \sqsubseteq)$ と $(Y, \sqsubseteq)$ について、関数 $f : X \to Y$ が**連続**であるとは、

1. $x_1 \sqsubseteq x_2$ なら $f(x_1) \sqsubseteq f(x_2)$
2. 可算単調列 $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ について　$f(\bigsqcup x_i) = \bigsqcup f(x_i)$

が成り立つことをいう。

（CPO を適当に位相空間とみなした場合の関数の連続性と同値）

最小不動点定理

定理 CPO $(X, \sqsubseteq)$ **上の連続関数** $f : X \to X$ **について、** $f(x) = x$ **となる** $x \in X$ **で最小のものが存在し、それは**

$$\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq f^3(\bot) \sqsubseteq \ldots$$

の上限 $\bigsqcup f^n(\bot)$ **で与えられる。**

証明 f **の連続性から**

$$f(\bigsqcup f^n(\bot)) = \bigsqcup f^{n+1}(\bot) = \bigsqcup f^n(\bot)$$

また、 $f(x) = x$ **とすると、** $\bot \sqsubseteq x$ **より** $f^n(\bot) \sqsubseteq f^n(x) = x$ **なので、** $\bigsqcup f^n(\bot) \sqsubseteq x$ **である。**

最小不動点による再帰プログラムの解釈

$[\mathbf{N} \rightharpoonup \mathbf{N}]$ 上の連続関数 F を

$$F(f)(x) \equiv \text{if } x = 0 \text{ then } 1 \text{ else } x \times f(x-1)$$

で定めると、$F^n(\bot) : \mathbf{N} \rightharpoonup \mathbf{N}$ は

$$F^n(\bot)(x) = \begin{cases} x \cdot (x-1) \cdot \ldots \cdot 2 \cdot 1 & x < n \\ \textbf{未定義} & x \geq n \end{cases}$$

である。したがって、F の最小不動点 $\bigsqcup F^n(\bot)$ は、階乗を計算する関数に他ならない。つまり、再帰プログラム

$$\mathsf{fact}(x) \equiv \text{if } x = 0 \text{ then } 1 \text{ else } x \times \mathsf{fact}(x-1)$$

の解釈を与えている。

領域理論の成功

以上では、領域理論のごく初歩的な事柄しか使っていない

しかし、この再帰プログラムを最小不動点によって特徴付けるという発想は、プログラミング言語の意味論のもっとも重要な洞察のひとつ

再帰データ型に対応する領域の構成というより困難な問題も、最小不動点の構成を拡張することによって非常にエレガントに解くことができる

再帰プログラムがプログラムの上の演算の不動点として捉えられるのに対し、再帰データ型は，プログラムの集まりの上の演算の不動点として捉えることができる

一般には後者のほうが困難であるが、適切な条件のもとではこれらのふたつの問題は同値となる

（領域理論はそのような条件を満たす構造の研究としても理解できる）

III. 再帰プログラムの幾何

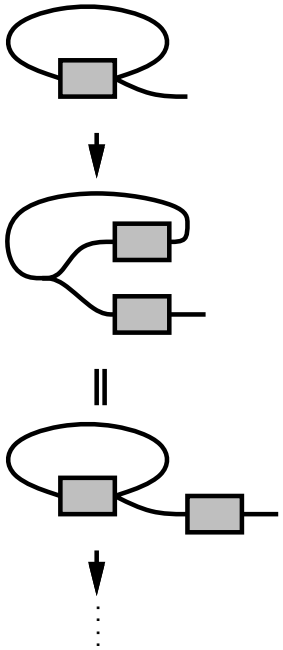
背景（１） 不動点意味論への不満

不動点意味論はシンプルでエレガントな数学理論だが、
計算の重要な側面を過度に単純化している

- どのように再帰計算が生み出されるか説明できない
- 再帰と、プログラミング言語の他の機能との相互作用が説明できない

必要なもの：これらの問題について調べるための適度な抽象化

巡回構造から生み出される再帰　（１９７０年代半ば～）

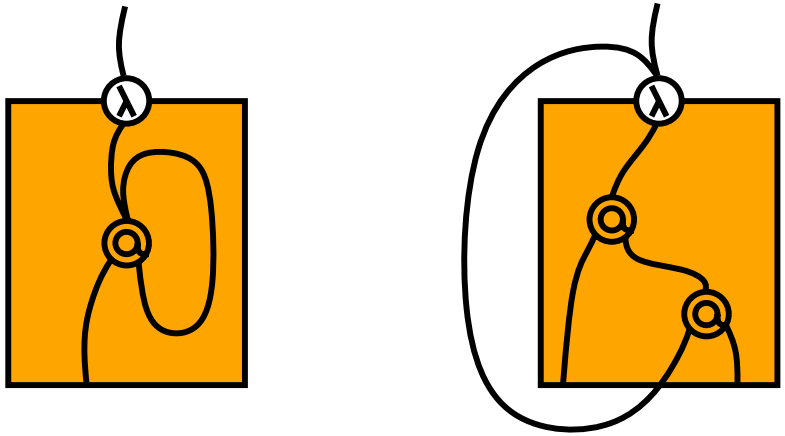


再帰呼び出し ＝ 巡回的に共有された資源のコピー

必要なもの：再帰計算と巡回構造の関係を正確に説明できる意味論

例：巡回的ラムダ計算

巡回的ラムダ計算における再帰演算子のふたつの実装



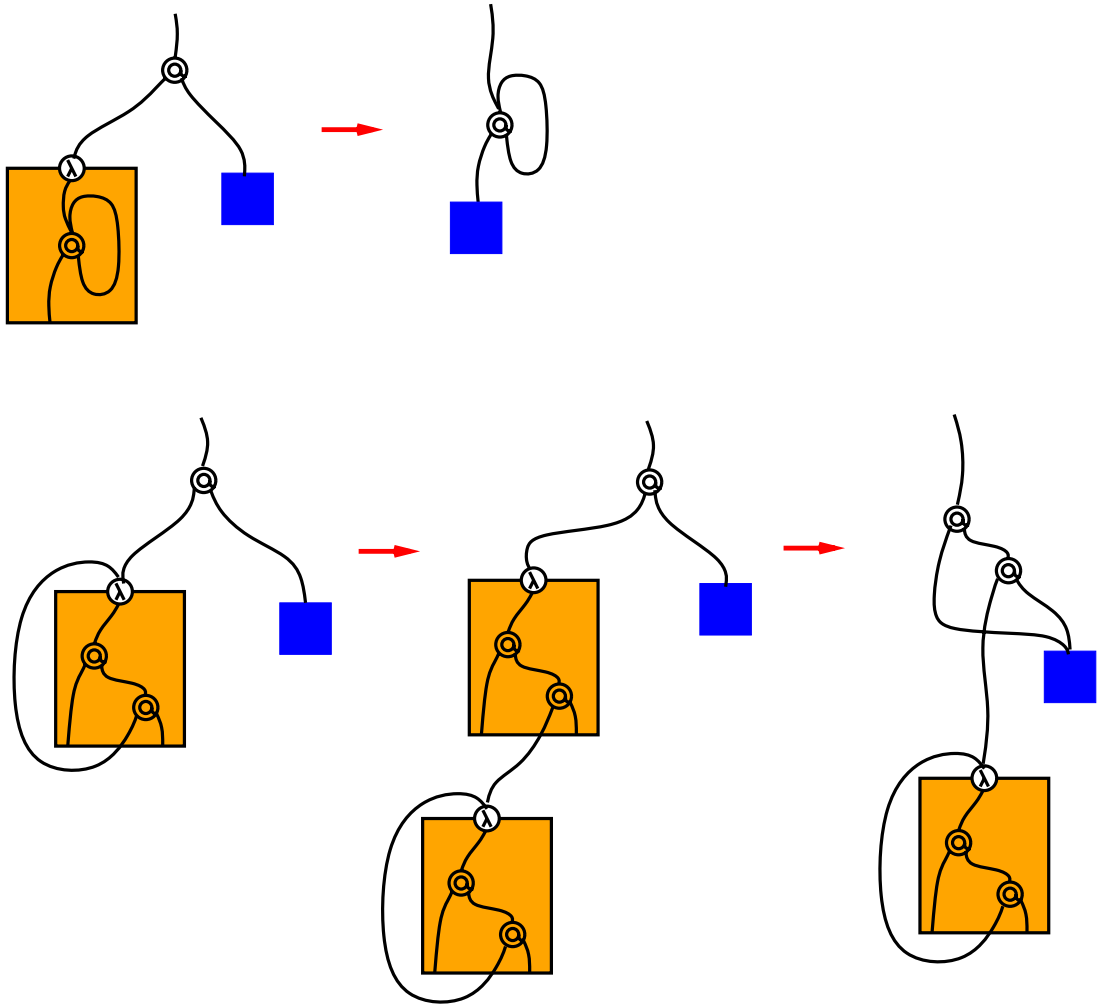
$\lambda f.\mathsf{letrec}\ x\ \mathsf{be}\ fx\ \mathsf{in}\ x \quad \stackrel{?}{=} \quad \mathsf{letrec}\ F\ \mathsf{be}\ \lambda f.f(Ff)\ \mathsf{in}\ F$

カリーの再帰演算子

$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$

チューリングの再帰演算子

$\Theta = Y(\lambda F.\lambda f.f(Ff))$



背景（2） 再帰プログラムの「幾何」?

再帰プログラムについて成り立つ「法則」

相互再帰（dinaturality）：

letrec x=g(f(x)) in x = letrec y=f(g(y)) in g(y)

対角化（diagonal property）：

letrec x={letrec y=h(x,y) in y} in x = letrec z = h(z,z) in z

同時再帰（Bekič property）：

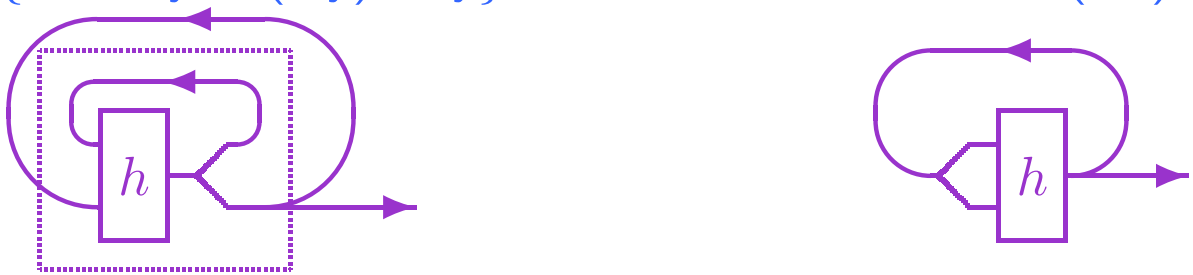
letrec x=f(x,y), y=g(x,y) in x = letrec x=f(x,{letrec y=g(x,y) in y}) in x

(dinaturality + diagonal $\Longrightarrow$ Bekič)

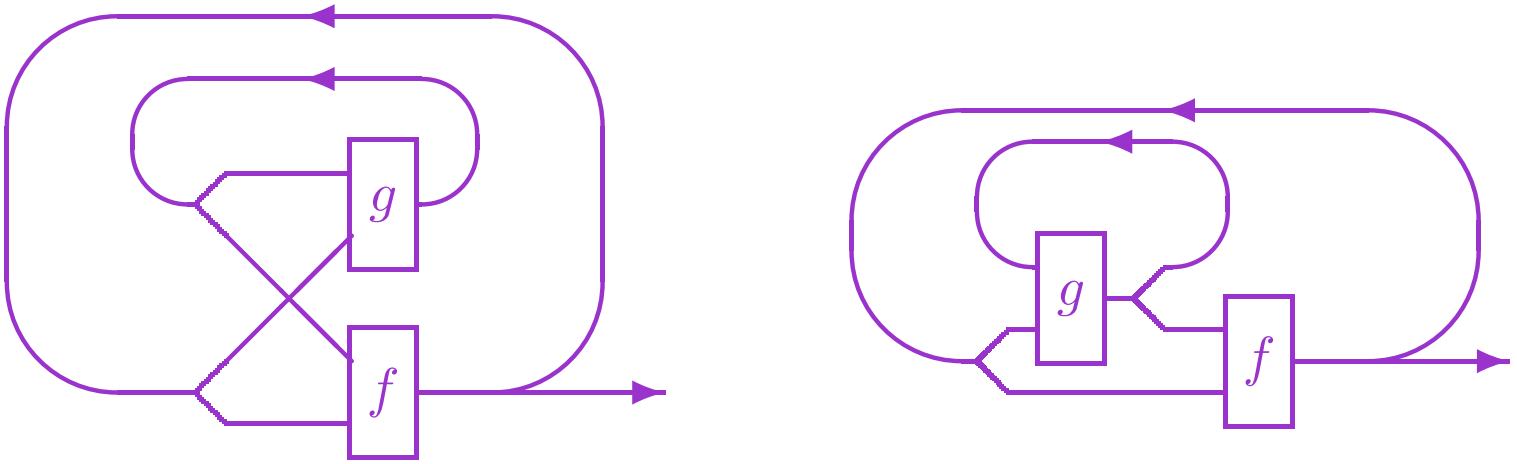
これらの法則は、幾何的な直観に対応している：

letrec x=g(f(x)) in x = letrec y=f(g(y)) in g(y)

letrec x={letrec y=h(x,y) in y} in x = letrec z = h(z,z) in z



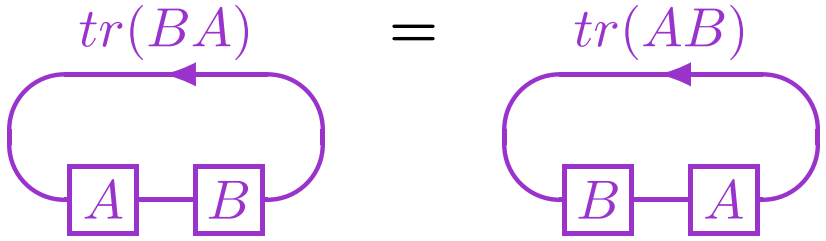
letrec x=f(x,y), y=g(x,y) in x = letrec x=f(x,{letrec y=g(x,y) in y}) in x



脱線：線型代数

正方行列 $A = (a_{i,j})$ **について、対角和** $tr(A) = \Sigma_i a_{i,i}$ **を** A **のトレースとよぶ**

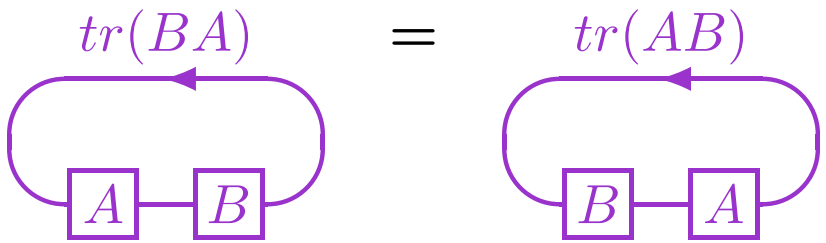
$m \times n$**-行列** A **と** $n \times m$**-行列** B **について** $tr(BA) = tr(AB)$ **がなりたつ**



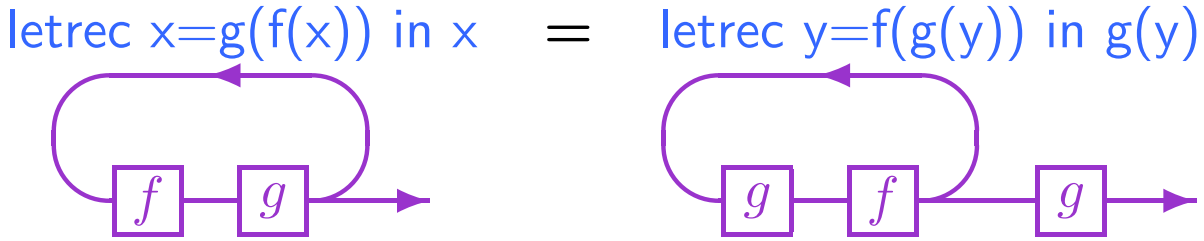
脱線：線型代数

正方行列 $A = (a_{i,j})$ について、対角和 $tr(A) = \Sigma_i a_{i,i}$ を A のトレースとよぶ

$m \times n$-行列 A と $n \times m$-行列 B について $tr(BA) = tr(AB)$ がなりたつ

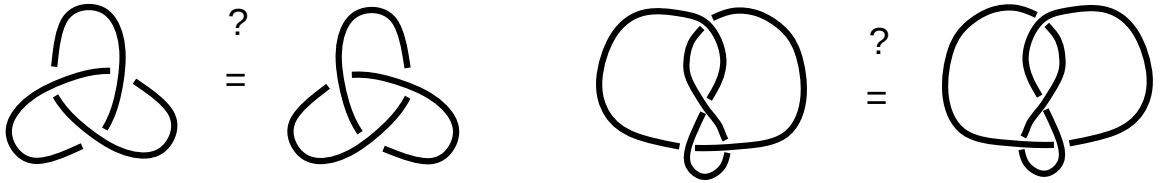


これは前述の再帰プログラムの「法則」と良く似ている：



結び目の理論との比較

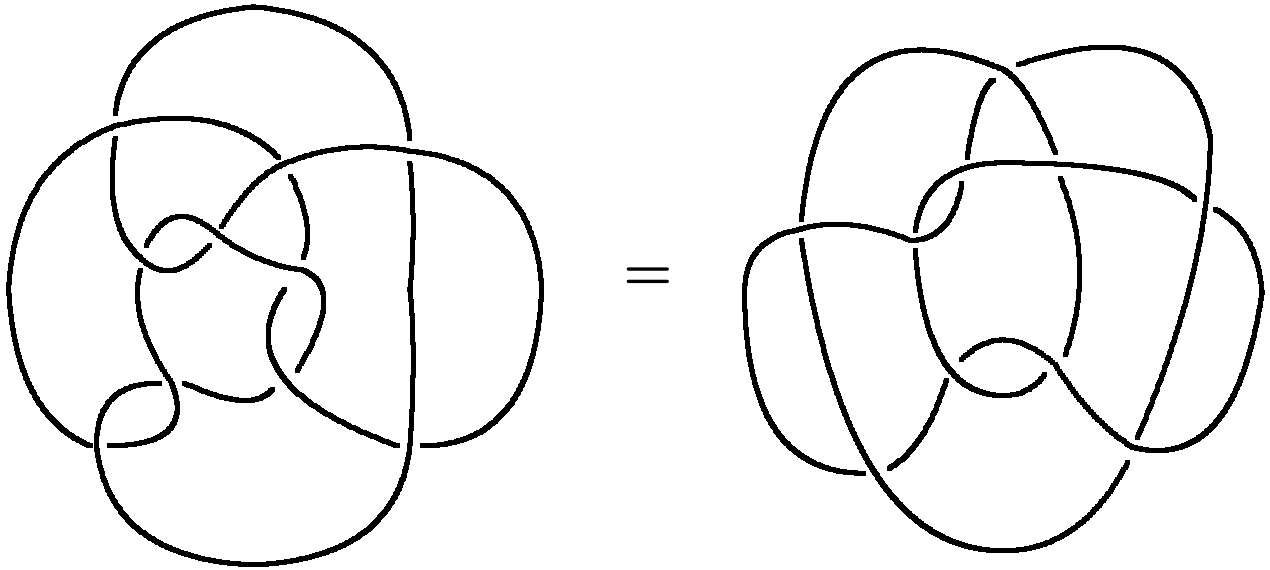
数学者の問題（結び目の分類問題）：
「これらのふたつの結び目は同じものだろうか？」



基本方針：結び目がもつ「不変な」数学的な量を考察
（結び目の不変量）

計算機科学者の問題：
「与えられたふたつのプログラムは同じ働きをするだろうか？」
基本方針：プログラムの実行に関して「不変な」抽象的な性質を考察
（プログラミング言語の意味論）

自明でない例



これらが同一の結び目であることがわかるまで、
およそ８０年かかっている

不変量の例：（２変数）ジョーンズ多項式　（１９８０年代半ば）

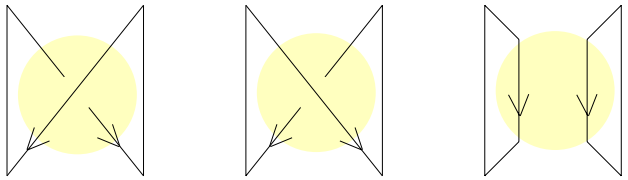
（向き付けられた）結び目を、環 $\mathbb{Z}[x, x^{-1}, y, y^{-1}]$ の要素（多項式）に対応させる、以下の性質を満たす写像 P が一意に存在：

(i) $L \sim L'$ なら $P(L) = P(L')$

(ii) P は自明な結び目を 1 にうつす

(iii) もしも (L_+, L_-, L_0) が　コンウェイの三つ組なら

$$xP(L_+) - x^{-1}P(L_-) = yP(L_0).$$



コンウェイの三つ組の例

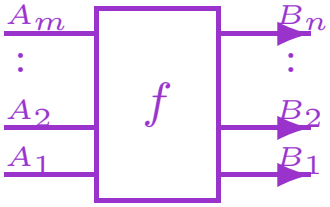
特に、L と L' が鏡像の関係にあれば、$P(L')(x, y) = P(L)(x^{-1}, y^{-1})$

たとえば、三つ葉の結び目とその鏡像は P によって区別される

モノイダル圏の幾何

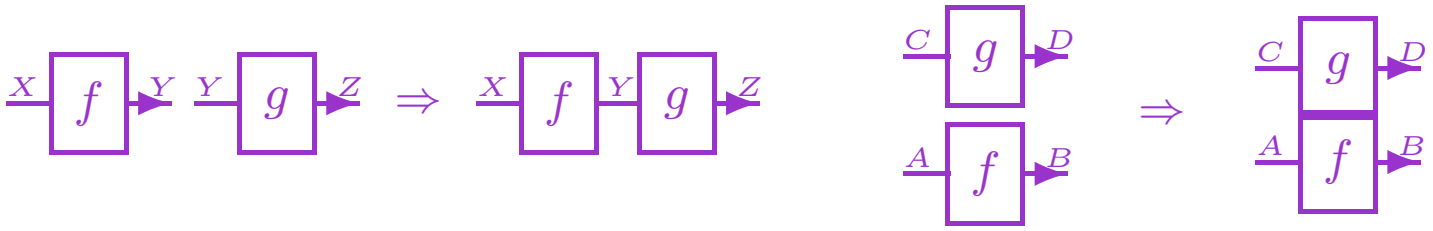
モノイダル圏 (テンソル圏)：テンソル積 $\otimes$ を持つ圏

モノイダル圏の射 $f : A_1 \otimes A_2 \otimes \ldots \otimes A_m \to B_1 \otimes B_2 \otimes \ldots \otimes B_n$ は



と図示できる

射の合成は逐次合成で、またテンソル積は平行合成で表現できる



また、任意のモノイダル圏について、このような図を用いた幾何的な推論は正しい（Geometry of Tensor Calculus, Joyal and Street**）**

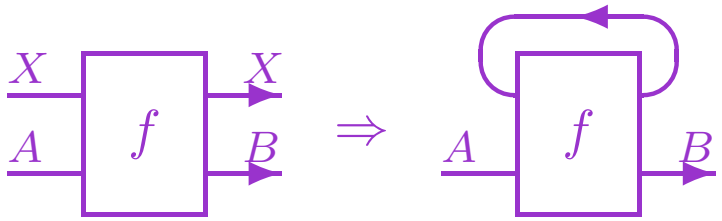
トレース付きモノイダル圏＝巡回構造の幾何

（Joyal, Street, Veiry １９９６）

結び目のような巡回的な構造のモデルとなる代数構造

以下のような「トレース演算子」をもつ（balanced な）モノイダル圏

$$\frac{f : A \otimes X \to B \otimes X}{Tr^X_{A,B}(f) : A \to B}$$



古典的な例：体 K 上の有限次元ベクトル空間と線型写像の圏
線型写像 $f : U \otimes_K W \to V \otimes_K W$ について、そのトレース
$Tr^W_{U,V}(f) : U \to V$ は以下の式で与えられる

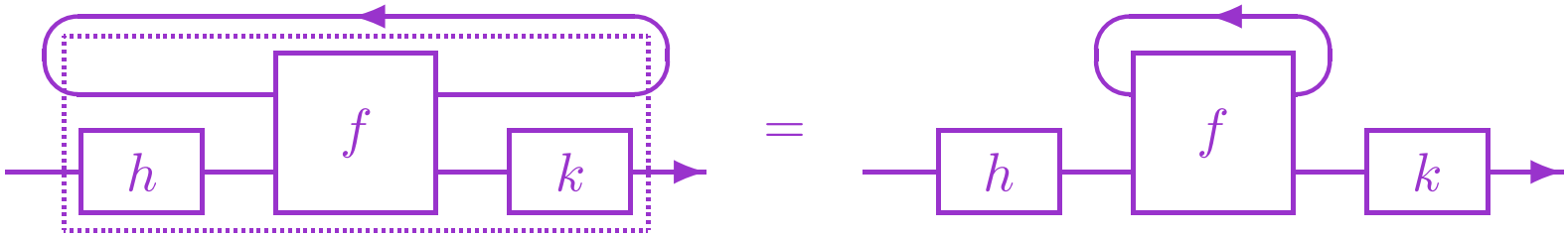
$$(Tr^W_{U,V}(f))_{i,j} = \Sigma_k f_{i\otimes k, j\otimes k}$$

最近の例：量子群の表現全体のなす圏（$\Longrightarrow$ 量子不変量）

トレース付きモノイダル圏の公理 *(1/2)*

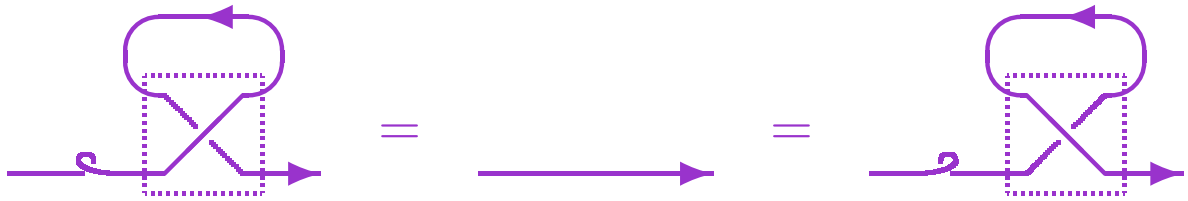
Tightening

$Tr^X_{A',B'}((k \otimes 1_X) \circ f \circ (h \otimes 1_X)) = k \circ Tr^X_{A,B}(f) \circ h$



Yanking

$Tr^X_{X,X}(c_{X,X}) \circ \theta^{-1}_X = 1_X = Tr^X_{X,X}(c^{-1}_{X,X}) \circ \theta_X$

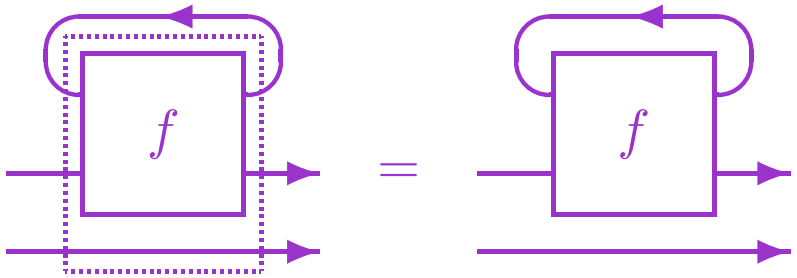


where $c_{X,Y} : X \otimes Y \xrightarrow{\simeq} Y \otimes X$ is a *braid*, while $\theta_X : X \xrightarrow{\simeq} X$ is a *twisting*

トレース付きモノイダル圏の公理 *(2/2)*

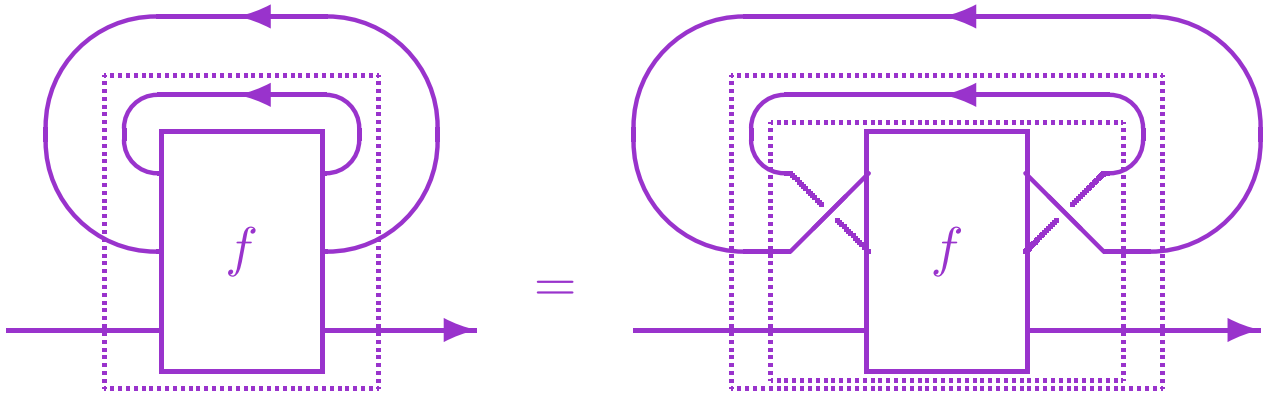
Superposing

$Tr^X_{C\otimes A, C\otimes B}(1_C \otimes f) = 1_C \otimes Tr^X_{A,B}(f)$



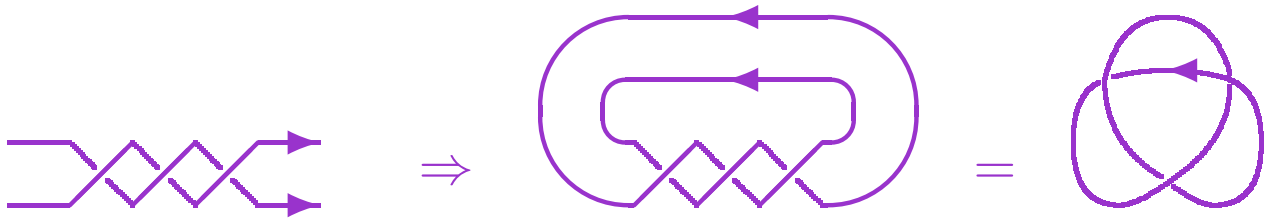
Exchange

$Tr^X_{A,B}(Tr^Y_{A\otimes X, B\otimes X}(f)) =$
$Tr^Y_{A,B}(Tr^X_{A\otimes Y, B\otimes Y}((1_B \otimes c_{Y,X}) \circ f \circ (1_A \otimes c^{-1}_{Y,X})))$

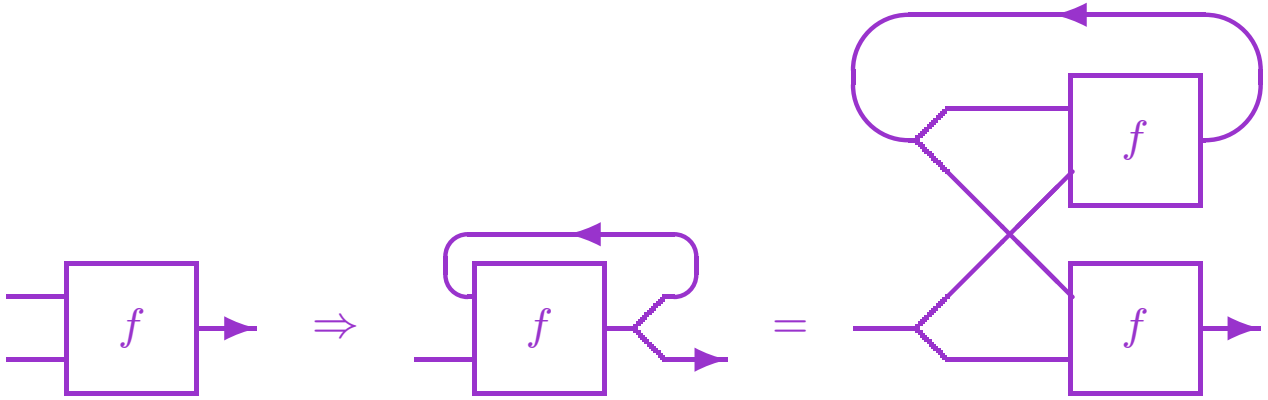


結び目も再帰もトレースを用いて理解できる

トレース（ブレイド閉包）を用いた三つ葉結び目の構成：



トレース（巡回共有構造）を用いた再帰計算：



再帰プログラムのトレースを用いた意味論　（１９９７〜）

適当な（古典的な）条件のもとでは、トレースによる意味論は不動点意味論と完全に一致する：

定理 (Hyland / **長谷川**)　**テンソル積** $\otimes$ **が直積（カルテジアン積）に制限された場合には、トレースと、**Bekič property **を満たす不動点演算子との間に、一対一対応が存在する**

例：領域理論におけるトレース

$f : A \times X \to B \times X$ **のトレース** $Tr^X_{A,B}(f) : A \to B$ **は**

$$Tr^X_{A,B}(f)(a) \;=\; f_1(a, \bigsqcup (x \mapsto f_2(a,x))^n(\bot))$$

ただし $f_1 = \pi_1 \circ f : A \times X \to B,\;\; f_2 = \pi_2 \circ f : A \times X \to X$

反例：巡回的ラムダ計算（値しかコピーできない $\Rightarrow$ **テンソル積は直積ではない）— 次の一般化でカバーされる**

再帰プログラムのトレースを用いた意味論　（続き）

反例：巡回的ラムダ計算— 以下の一般化でカバーされる：

定理 (長谷川)　直積を持つ圏と、トレース付きモノイダル圏の間のモノイダル随伴関手から、dinaturality を満たす不動点演算子が構成できる

直積を持つ圏	＝	値の世界（コピーができる）
トレースを持つ圏	＝	巡回的な計算の世界
モノイダル随伴関手	＝	値の世界と計算の世界を結びつける

この定理によって、巡回的ラムダ計算における再帰計算の、トレースによる意味論が展開できる

巡回共有構造から生まれる再帰 $\Rightarrow$ 線型不動点

先の定理の内容を少し詳しく述べると：

有限直積を持つ圏 $\mathbf{C}$ と、トレースつきモノイダル圏 $\mathbf{D}$、および左随伴モノイダル関手 $F : \mathbf{C} \to \mathbf{D}$ を考える。このとき、$\mathbf{D}$ の射 $f : FA \otimes X \longrightarrow X$ について、

$$FA \xrightarrow{f^\dagger} X \quad = \quad FA \xrightarrow{F\Delta} F(A \times A) \xrightarrow{\simeq} FA \otimes FA \xrightarrow{1_{FA} \otimes f^\dagger} FA \otimes X \xrightarrow{f} X$$

をみたす $f^\dagger : FA \longrightarrow X$ が存在する

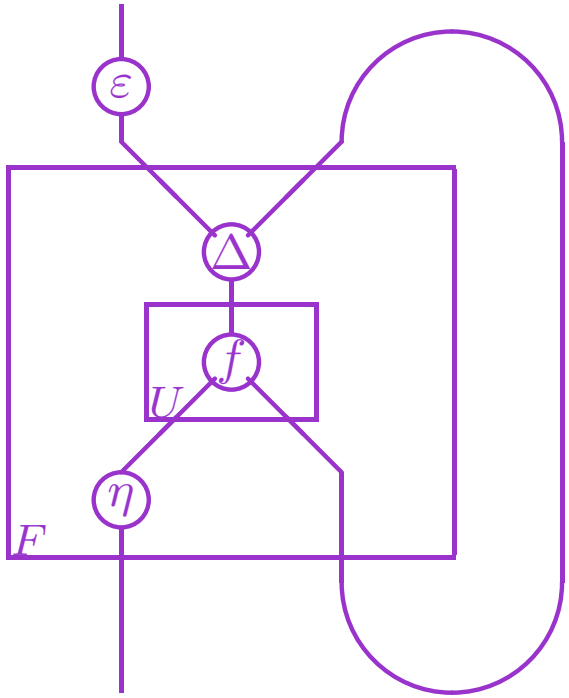
Girard の線型論理に対応するラムダ計算に翻訳すると：

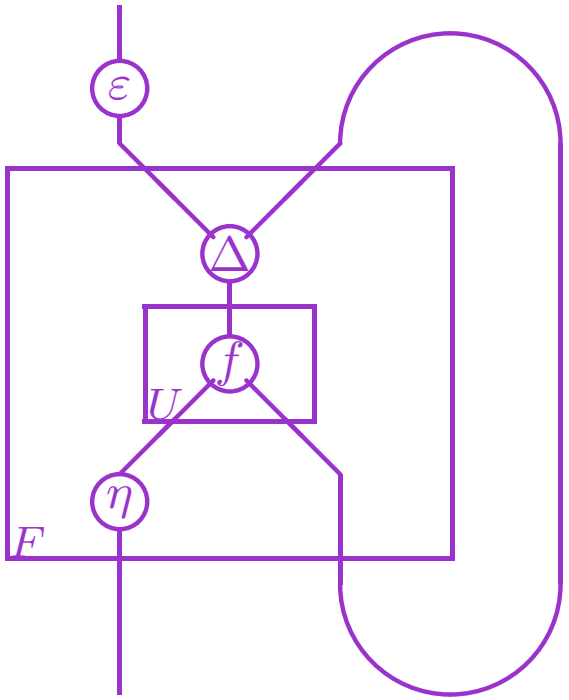
$$\mathsf{Y}\,(!f) \;=\; f\,(\mathsf{Y}\,(!f)) \qquad (f : X \multimap X)$$

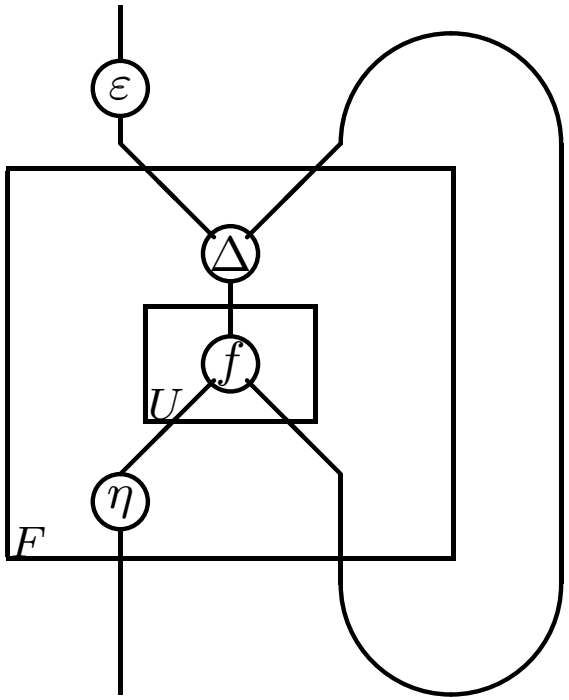
をみたす（線型）不動点演算子 $\mathsf{Y} : !(X \multimap X) \multimap X$ が存在する

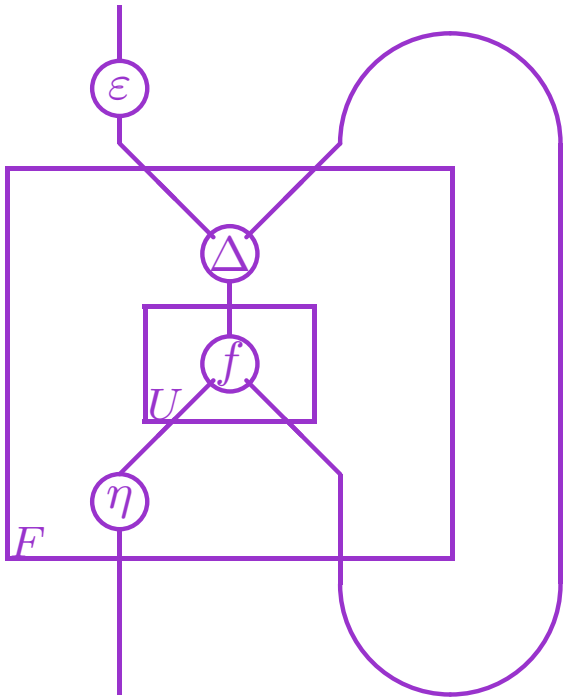
この $f^\dagger$ は以下のようにあらわされる

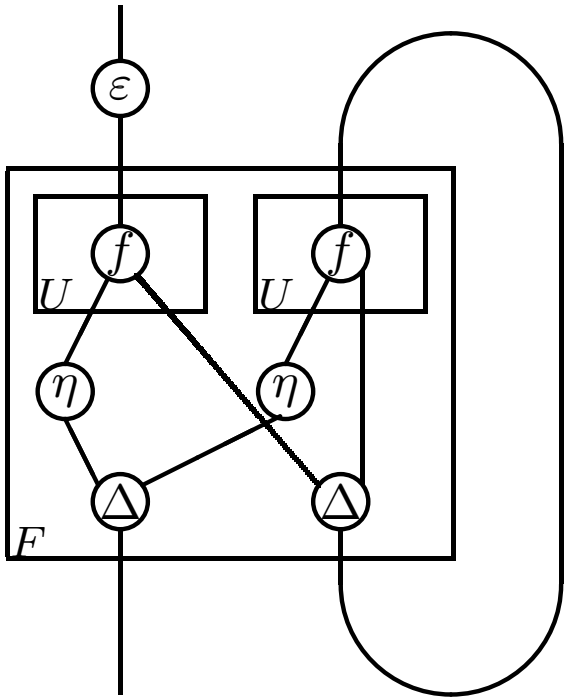
（Mellies: “Functorial boxes”, CSL’06**）：**

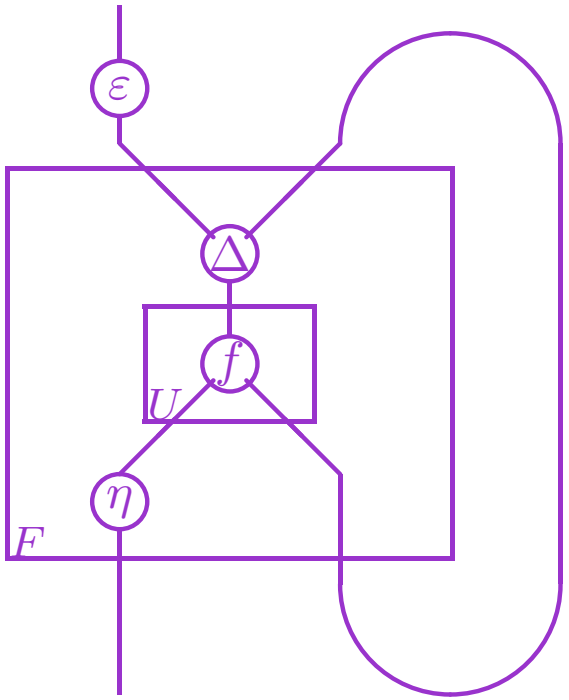


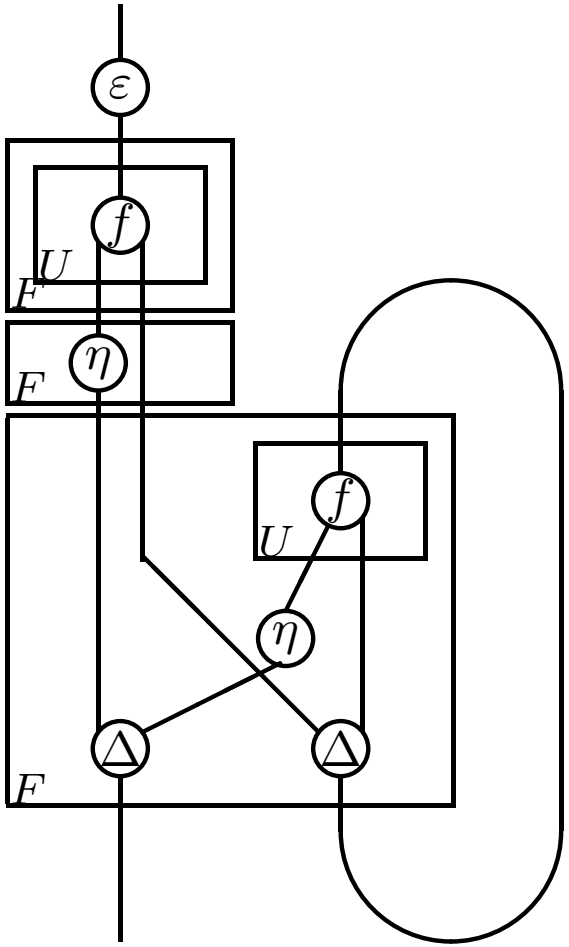

ε
Δ
f
U
η
F

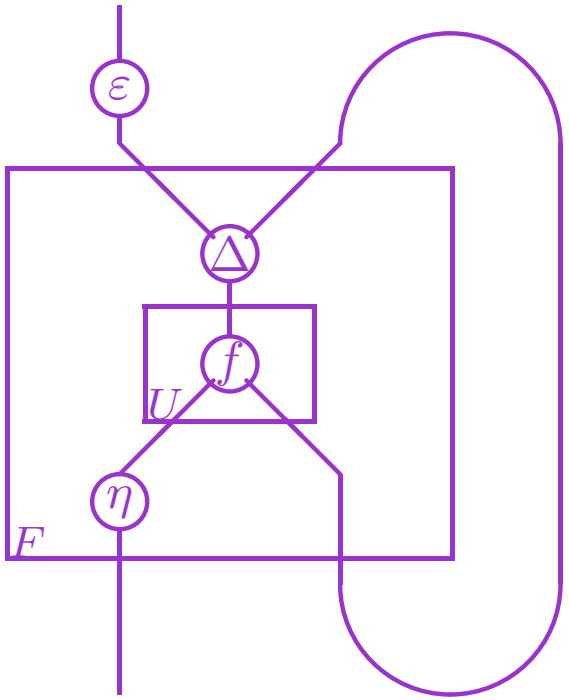

ε
Δ
f
U
η
F

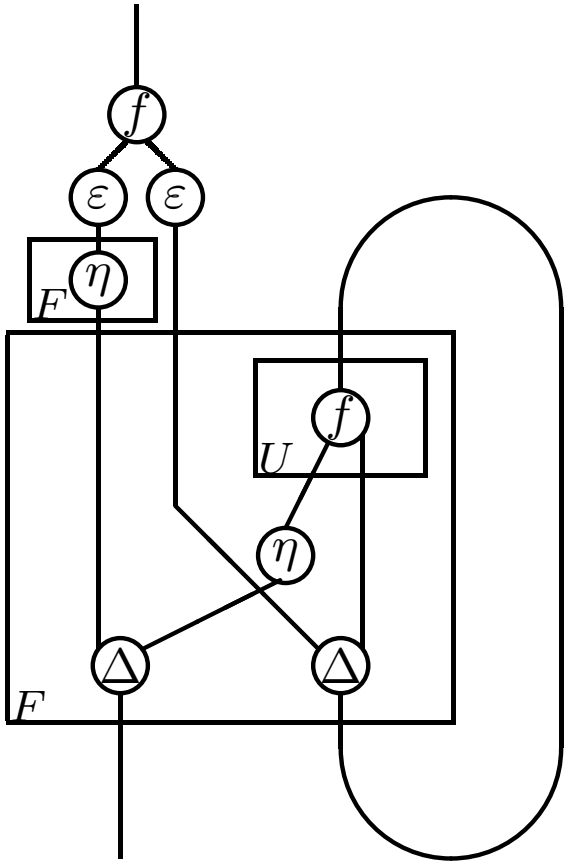

ε
Δ
f
U
η
F

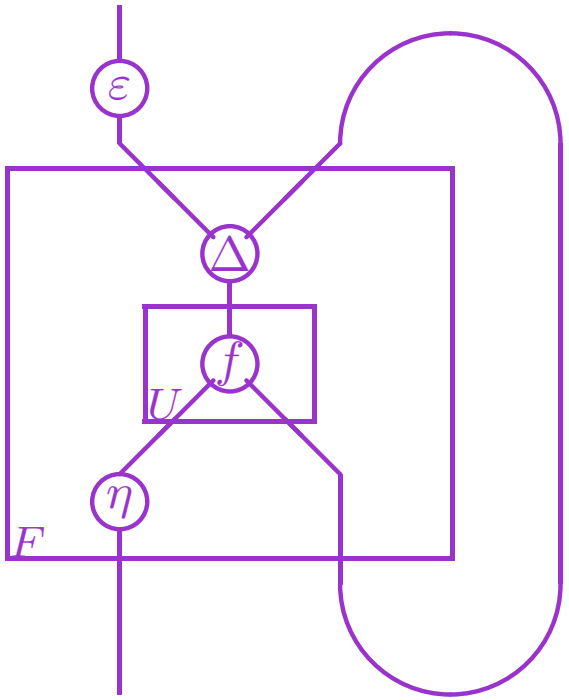

ε
f
f
U
U
η
η
Δ
Δ
F

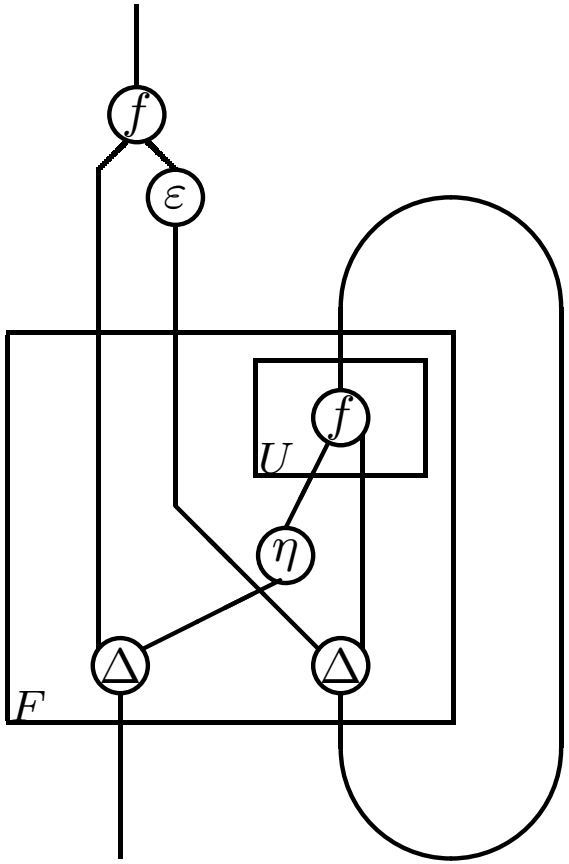

ε
Δ
f
U
η
F

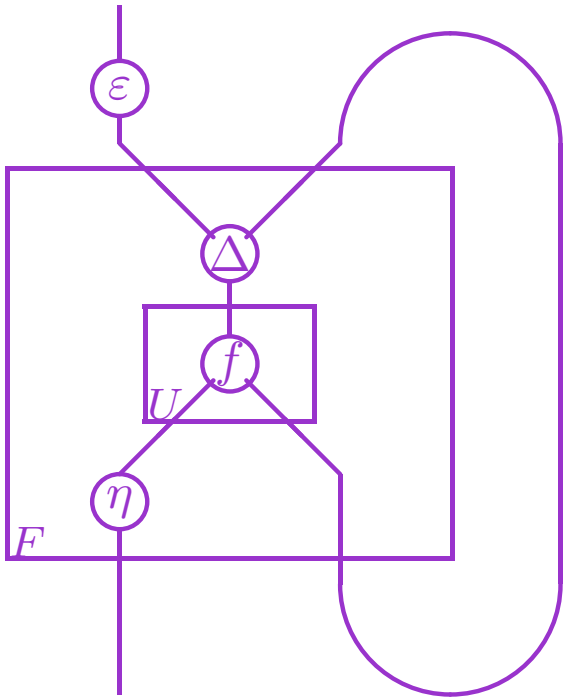

ε
f
U
F
η
F
f
U
η
Δ
Δ
F

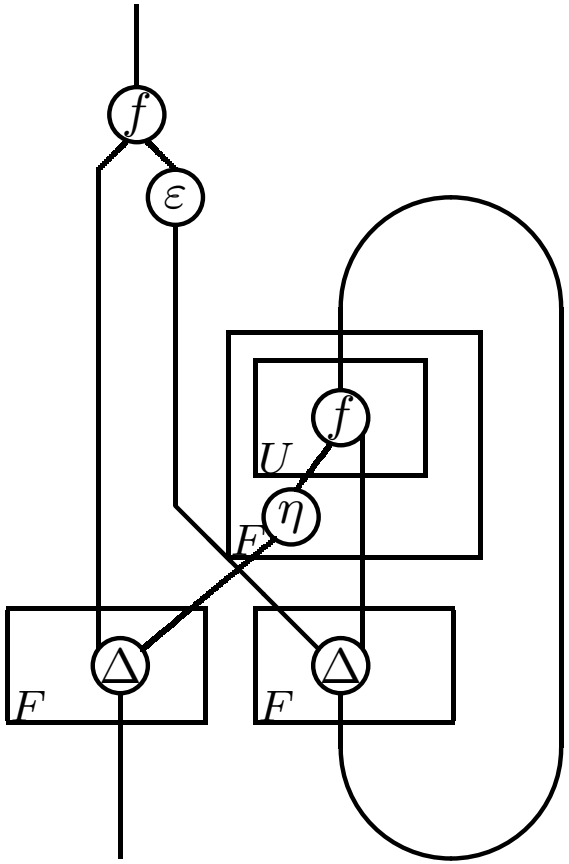

ε
Δ
f
U
η
F

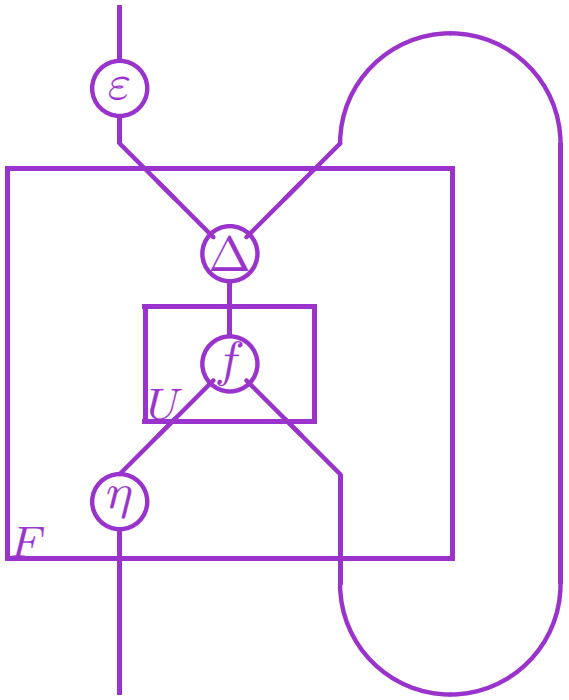

f
ε
ε
η
F
f
U
η
Δ
Δ
F

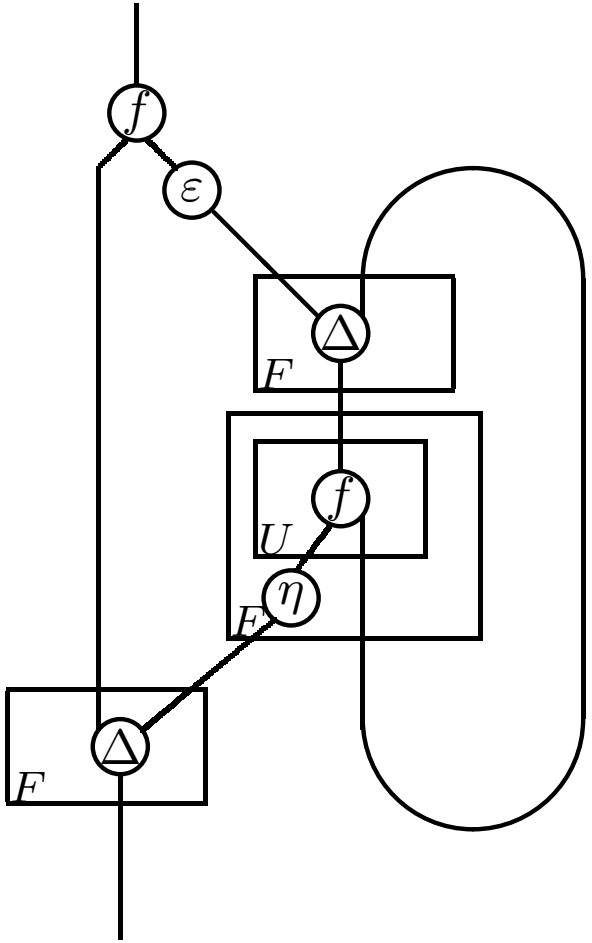

ε
Δ
f
U
η
F

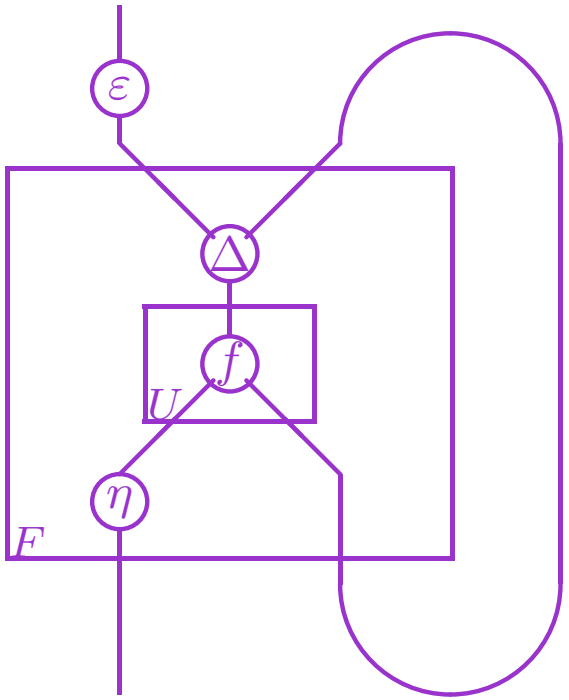

f
ε
f
U
η
Δ
Δ
F

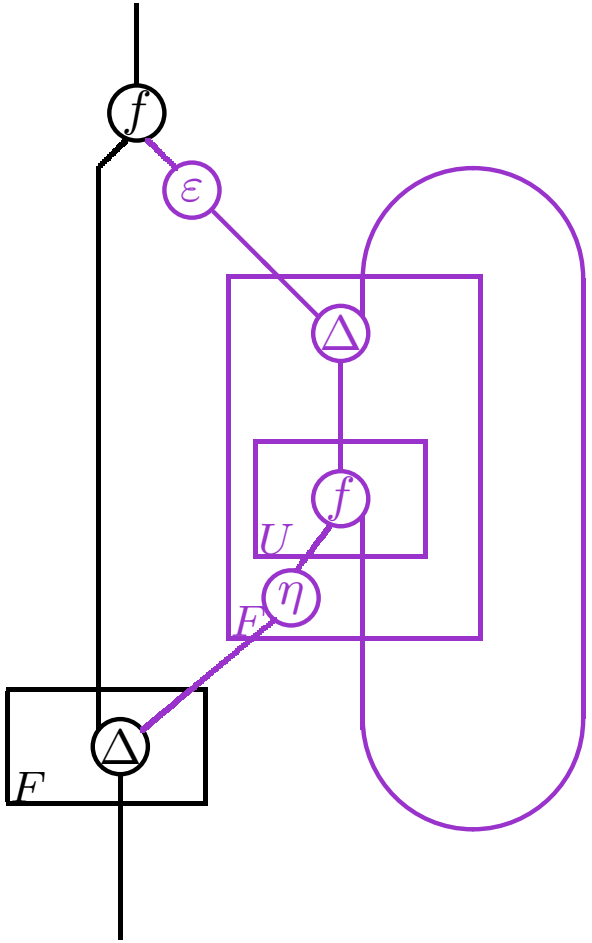

ε
Δ
f
U
η
F


f
ε
f
U
η
F
Δ
Δ
F
F


ε
Δ
f
U
η
F

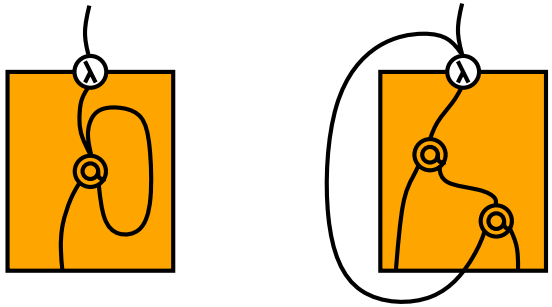

f
ε
Δ
F
f
U
η
F
Δ
F


ε
Δ
f
U
η
F


f
ε
Δ
f
U
η
F
Δ
F

例：巡回的共有構造の解釈

非決定性計算のなすトレース付きモノイダル圏では



$\mathsf{fix1} = \lambda f.\mathsf{letrec}\ x\ \mathsf{be}\ fx\ \mathsf{in}\ x$ $\qquad$ $\mathsf{fix2} = \mathsf{letrec}\ F\ \mathsf{be}\ \lambda f.f(Ff)\ \mathsf{in}\ F$

$[\![\mathsf{fix1}\ M]\!] = \bigcup_{f \in [\![M]\!]} \{x \mid (x,x) \in f\}$ $\qquad$ $[\![\mathsf{fix2}\ M]\!] = \bigcup_{f \in [\![M]\!]} \bigcup_{f \circ A = A} A$

これらのふたつの再帰演算子は、この「不変量」によって明確に区別される（非決定性プログラミング言語において異なる振る舞いをする）

$$\mathsf{fix1}\,(\lambda x.(0\ \mathsf{or}\ x+1)) \quad = \quad 0\ \mathsf{or}\ \bot$$

$$\mathsf{fix2}\,(\lambda x.(0\ \mathsf{or}\ x+1)) \quad = \quad 0\ \mathsf{or}\ 1\ \mathsf{or}\ 2\ \mathsf{or}\ \ldots\ \mathsf{or}\ \bot$$

トレースによる意味論の発展　（１９９７～現在）

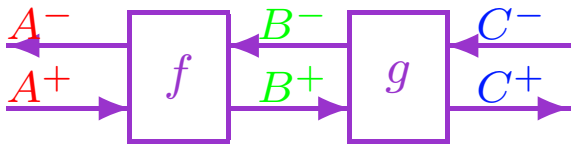
この、「巡回構造から生じる再帰計算の不変量」の理論は、従来の
「再帰計算の不動点意味論」を真に拡張したものとなっている。
逆に、従来の不動点意味論は、トレースによる一般的な意味論の
特殊な場合として明快に理解できる。

さらに、トレース付きモノイダル圏をもとにした、新しいモデルが、
様々な研究者によって発見され、調べられている

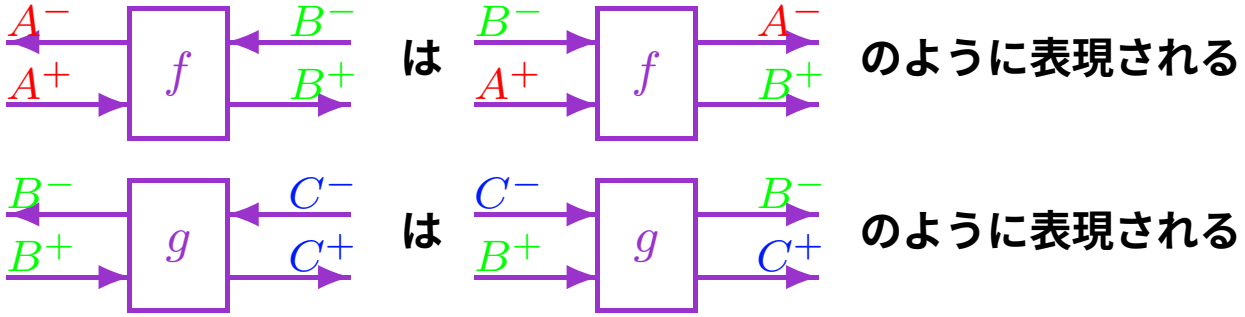
最近の展開：
トレース付きモノイダル圏における一様性原理の発見とその応用、
公理的領域理論への応用、相互作用の幾何との関係、
再帰と他の制御構造の関係の分析、線型な型体系における再帰の分析
など

双方向のコミュニケーションのモデル

相互作用的な（双方向の）コミュニケーションのモデルを与えることを考える:



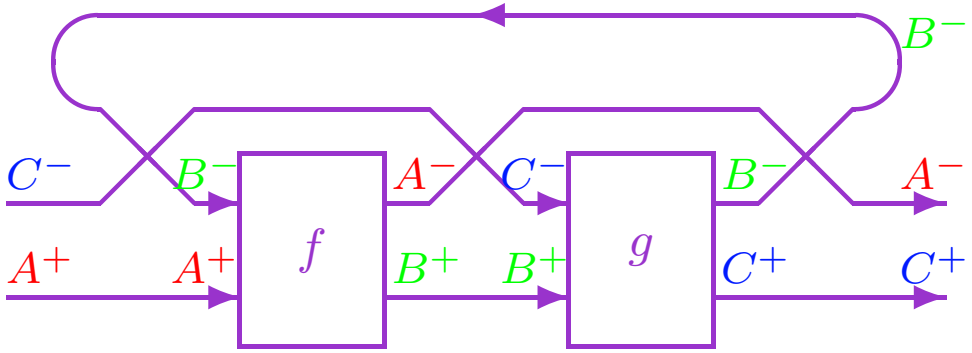
モノイダル圏では:



しかし、これらをどうやって合成したらよいのか？

Geometry of Interaction （相互作用の幾何）(Girard/Abramsky)

解決策: トレースを使えばよい！



**これは、トレース付きモノイダル圏から自己双対な圏を
つくりだす普遍的な構成を与えている**

応用：平行計算、線型論理、古典論理のさまざまなモデルの構成

応用：線型不動点のモデルの構成

**最近の応用：プログラム変換における不要な高階中間コードの除去
（勝股＆西村、ICFP'06）**

Part IV. **まとめ**

まとめ

「再帰プログラムの幾何」は「再帰プログラムの不動点意味論」の一般化になっている

これはアドホックな一般化ではなく、再帰計算のもつ本質的な巡回的な構造を、トレースという数学構造を土台に浮彫りにしたものである

逆に、「再帰プログラムの幾何」の一般化された立場からみることで、領域理論に基づく不動点意味論の長所も短所も明確になる

「再帰プログラムの幾何」の展望

「再帰プログラムの幾何」の主張

再帰プログラムの意味論を与えること＝
適切な条件をみたすトレース付きモノイダル圏を与えること

現在の主な課題：

- 具体的なトレース付きモノイダル圏を見つけるのは決して易しくない（モデルを構成する技法の拡充が必要）
- 「再帰データ型の幾何」？
- 「線型不動点」から通常の不動点を得られるかどうかは自明でない（cf. 線型論理での $A \Rightarrow B \simeq !A \multimap B$）

  線型不動点演算子の型：$\forall X.!(X \multimap X) \multimap X$

  不動点演算子の型：$\forall X.!(!X \multimap X) \multimap X$

  前者から後者が得られるための条件はまだわかっていない

再帰プログラムの意味論の展望

(cf. Recursive Programs in the Abstract - PPL2004)

- **他の制御構造や副作用を伴うプログラムとの相互作用の分析**
  継続＋繰り返し $\Rightarrow$ 再帰（Filinski '94**)、値呼び不動点演算子の公理と意味論（長谷川-角谷** '01**）**
- **副作用を伴う再帰プログラムの幾何** （Jeffrey, Erkok-Launchbury, Benton-Hyland **...**）
- **パラメトリックポリモルフィズムと再帰（線型パラメトリシティ:** Plotkin, Pitts, **長谷川立 ...）（第一級継続への拡張：長谷川** '05**、モナディックな副作用への拡張：**Simpson '06**）**
- **一様性（**Plotkin **の公理：長谷川**'02**、角谷-長谷川**'03**）**
- **再帰と繰り返しの双対性（角谷**'02**）**

「プログラミング言語の意味論」の展望

再帰を含め、プログラムの制御構造の明快な意味論を展開するには、以下の点が重要

- algebraic（代数的）：
  等式で書き下せる理論であること
- uniformity（一様性）：
  様々な状況や制約に対し普遍的な性格をもつ理論であること

前者は 90 年代以降の計算の意味論の流れのキーワードでもある

後者は最近の一連の成果（特に最近のプログラムの多相性と制御構造に関する研究）において顕著

プログラムの制御構造のための一般論は、
適切な一様性を満たす代数的理論として構築されるべき

結び

プログラミング言語の意味論の研究は、ありのままに
プログラムをみない手法を身につけるための修行である

表層にとらわれないことによって浮き彫りになってくる本質に
目を凝らすことが大切

ありがとうございました