# プログラミング言語の意味論と圏論

長谷川真人
京都大学数理解析研究所

**概 要**

プログラミング言語の意味論（プログラム意味論）は、プログラムの構造を数学的に定式化・抽象化することによってコンピュータソフトウェアに関する諸問題を解決することを目的とする、コンピュータサイエンスの一分野です。そこでは、抽象的な現代数学の象徴とも言われる圏論が活発に応用されてきており、今や圏論抜きにプログラム意味論を語ることは困難といっても過言ではありません。

この講義では、圏論を用いたプログラム意味論の概要を紹介しつつ、そもそもプログラム意味論に求められる数学とはどのようなものなのか、なぜ圏論なのか、という素朴な疑問に、私なりに答えてみたいと思います。

## 1　プログラミング言語の意味論とはなにか

プログラミング言語の意味論とは、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ことを目的とする研究分野です。その応用として、プログラムの見通し良い設計、プログラムの性質の検証、およびプログラムの効率化・最適化などがあげられます。しかし、ほとんどの方にとって、意味論というのはあまり馴染みのない言葉ではないかと思いますし、それが数学とどのように関係するのか、想像することも困難かも知れません。そこで、まずは、そもそもプログラミング言語の意味論とはどのような分野なのかについて、基本的なところから解説していきたいと思います。

### 1.1　プログラムの意味とは

プログラミング言語のプログラムは、コンピュータに何かをさせるための指示を記述する文章であり、通常、有限長の記号の列として与えられます。しかし、記号列にはじめからなにか固有の意味が備わっているわけではもちろんなく、プログラムの表現する計算＝プログラムの**意味**（semantics）については、別に定義する必要があります。この、プログラムの意味を数学的に定義すること、およびその理論的な枠組みのことを、**プログラミング言語の意味論**（**プログラム意味論**）と呼びます。

プログラミング言語の意味論にはいくつかの種類があります。ここでは、その概要を、とても単純なプログラミング言語を例にとって説明します。

**簡単な言語**　真偽値を計算する、とても簡単な（関数型）プログラミング言語 **FB** を与えます。準備として、変数をあらわす記号 $\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots$ をあらかじめ用意しておきます。**FB** のプログラムは、以下のようなものです。

1. 変数記号は **FB** のプログラムである。
2. 記号 $\mathsf{T}$ と $\mathsf{F}$ は **FB** のプログラムである。
3. $P$ が **FB** のプログラムであるとき、$(\mathsf{not}\, P)$ は **FB** のプログラムである。

4. $P_1$ と $P_2$ が **FB** のプログラムであるとき、$(P_1 \mathsf{\ and\ } P_2)$ は **FB** のプログラムである。

5. $x$ が変数記号、$P_1$ と $P_2$ が **FB** のプログラムであるとき、$(\mathsf{let}\, x \,\mathsf{be}\, P_1 \,\mathsf{in}\, P_2)$ は **FB** のプログラムである。

6. 以上の規則で得られるものだけが、**FB** のプログラムである。

このように、どのような記号列が正しいプログラムであるかを定める規則＝文法（とその理論的枠組み）のことを、プログラミング言語の**構文論**（syntax）と言います。

たとえば、以下のような記号列は **FB** のプログラムです。

$$(\mathsf{let\ x\ be\ (T\ and\ y)\ in\ ((not\ x)\ and\ F)})$$

プログラム $P$ の中の**束縛されていない変数（自由変数）**の集まり $fv(P)$ を、

- $fv(x) = \{x\}$ （ただし $x$ は変数記号）
- $fv(\mathsf{T}) = fv(\mathsf{F}) = \emptyset$
- $fv((\mathsf{not}\, P)) = fv(P)$, $fv((P_1 \mathsf{\ and\ } P_2)) = fv(P_1) \cup fv(P_2)$
- $fv((\mathsf{let}\, x \,\mathsf{be}\, P_1 \,\mathsf{in}\, P_2)) = fv(P_1) \cup (fv(P_2) - \{x\})$

と定め、$fv(P) =$ となるような $P$ のことを**閉じたプログラム**と呼ぶことにします。閉じたプログラムは、定義が不明な変数がないプログラムのことだ、とも言えます。

**FB** の閉じたプログラムの意味は、直感的にはほぼ自明だと思います。たとえば、プログラム $(P_1 \mathsf{\ and\ } P_2)$ は、$P_1$ の計算結果と $P_2$ の計算結果の論理積を求めるものです。また、$(\mathsf{let}\, x \,\mathsf{be}\, P_1 \,\mathsf{in}\, P_2)$ は、$P_1$ の計算結果を $P_2$ 中の $x$ にあてはめ、然る後に $P_2$ の計算を行うもの、と考えて差し支えありません。

## 1.2 操作的意味論

実際にプログラミング言語の処理系が行っているような具体的な計算の規則によってプログラムの意味を与える流儀は、**操作的意味論**（operational semantics）と呼ばれます。操作的意味論が与えられると、コンピュータ上で動作するプログラミング言語の処理系を与えることが（少なくとも原理的には）できるので、現在では、操作的意味論は、プログラミング言語の定義を与える有力な手法とみなされています。

**FB** の操作的意味論は、以下のように定められます。閉じたプログラム $P$ に対し、その実行結果が $V \in \{\mathsf{T}, \mathsf{F}\}$ であることを、$P \Downarrow V$ と書くことにします。

1. $\mathsf{T} \Downarrow \mathsf{T}$ である。また、$\mathsf{F} \Downarrow \mathsf{F}$ である。

2. $P \Downarrow \mathsf{T}$ ならば $(\mathsf{not}\, P) \Downarrow \mathsf{F}$ である。$P \Downarrow \mathsf{F}$ ならば $(\mathsf{not}\, P) \Downarrow \mathsf{T}$ である。

3. $P_1 \Downarrow \mathsf{T}$ かつ $P_2 \Downarrow \mathsf{T}$ ならば $(P_1 \mathsf{\ and\ } P_2) \Downarrow \mathsf{T}$ である。$P_1 \Downarrow \mathsf{T}$ かつ $P_2 \Downarrow \mathsf{F}$ ならば $(P_1 \mathsf{\ and\ } P_2) \Downarrow \mathsf{F}$ である。$P_1 \Downarrow \mathsf{F}$ ならば $(P_1 \mathsf{\ and\ } P_2) \Downarrow \mathsf{F}$ である。

4. $P_1 \Downarrow V_1$ かつ $P_2[x := V] \Downarrow V_2$ ならば $(\mathsf{let}\, x \,\mathsf{be}\, P_1 \,\mathsf{in}\, P_2) \Downarrow V_2$ である。ここで、$P[x := V]$ は、$P$ のなかの（自由な）変数 $x$ をすべて $V$ に置き換えて得られるプログラムである。

数理論理学の記法（導出木）を用いて、「$Z$ である」「$X$ ならば $Z$ である」「$X$ かつ $Y$ ならば $Z$ である」を

$$\frac{}{Z} \qquad \frac{X}{Z} \qquad \frac{X \quad Y}{Z}$$

と表記すると、操作的意味論の規則は

$$\frac{}{\mathsf{T} \Downarrow \mathsf{T}} \quad \frac{}{\mathsf{F} \Downarrow \mathsf{F}} \quad \frac{P \Downarrow \mathsf{T}}{(\mathsf{not}\, P) \Downarrow \mathsf{F}} \quad \frac{P \Downarrow \mathsf{F}}{(\mathsf{not}\, P) \Downarrow \mathsf{T}} \quad \frac{P_1 \Downarrow \mathsf{T} \quad P_2 \Downarrow \mathsf{T}}{((P_1\ \mathsf{and}\ P_2) \Downarrow \mathsf{T}}$$

$$\frac{P_1 \Downarrow \mathsf{T} \quad P_2 \Downarrow \mathsf{F}}{((P_1\ \mathsf{and}\ P_2) \Downarrow \mathsf{F}} \quad \frac{P_1 \Downarrow \mathsf{F}}{((P_1\ \mathsf{and}\ P_2) \Downarrow \mathsf{F}} \quad \frac{P_1 \Downarrow V_1 \quad P_2[x := V] \Downarrow V_2}{(\mathsf{let}\, x\, \mathsf{be}\, P_1\, \mathsf{in}\, P_2) \Downarrow V_2}$$

となります。たとえば、プログラム $(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})$ の操作的意味は、以下の導出木で表され、その実行結果は F となります。

$$\frac{\dfrac{\dfrac{}{\mathsf{F} \Downarrow \mathsf{F}}}{(\mathsf{not}\,\mathsf{F}) \Downarrow \mathsf{T}} \quad \dfrac{\dfrac{}{\mathsf{T} \Downarrow \mathsf{T}} \quad \dfrac{}{\mathsf{F} \Downarrow \mathsf{F}}}{(\mathsf{T}\ \mathsf{and}\ \mathsf{F}) \Downarrow \mathsf{F}}}{(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)}) \Downarrow \mathsf{F}}$$

これは、プログラムの一部を順次書き換えていく、以下のような計算ステップの列に対応しています。

$$(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)}) \to (\mathsf{let\,x\,be\,T\,in\,(x\,and\,F)}) \to (\mathsf{T}\ \mathsf{and}\ \mathsf{F}) \to \mathsf{F}$$

### 1.3 表示的意味論

一方、プログラムを、具体的な実行方法とは独立に、なんらかの数学的な対象に対応させて意味を定める、という流儀もあります。これを、**表示的意味論**（denotational semantics）と呼びます。

**FB** のプログラム $P$ の表示的意味は、$fv(P) \subseteq \Gamma$ であるような変数の集合 $\Gamma$ と以下の規則で定まる真偽値関数 $[\![P]\!] : \{0,1\}^\Gamma \longrightarrow \{0,1\}$ で与えられます。ただし、$\{0,1\}^\Gamma$ の要素を写像 $\rho : \Gamma \longrightarrow \{0,1\}$ で表すことにします。また、表示的意味論の慣習で、$[\![P]\!](\rho)$ を $[\![P]\!]_\rho$ と書くことにします。

$$\begin{array}{lcl}
[\![x]\!]_\rho & = & \rho(x) \\
[\![\mathsf{T}]\!]_\rho & = & 1 \\
[\![\mathsf{F}]\!]_\rho & = & 0 \\
[\![(\mathsf{not}\, P)]\!]_\rho & = & 1 - [\![P]\!]_\rho \\
[\![(P_1\ \mathsf{and}\ P_2)]\!]_\rho & = & [\![P_1]\!]_\rho \cdot [\![P_2]\!]_\rho \\
[\![(\mathsf{let}\, x\, \mathsf{be}\, P_1\, \mathsf{in}\, P_2)]\!]_\rho & = & [\![P_2]\!]_{\rho[x \mapsto [\![P_1]\!]_\rho]}
\end{array}$$

ただし最後の行の右辺で、$\rho[x \mapsto v]$ は、$x$ 以外の変数については $\rho$ が定める値を取り、$x$ については $v$ を値とする写像です。

特に $P$ が閉じたプログラム、すなわち $fv(P) = \emptyset$ の場合は、$[\![P]\!]_\rho$ の値は $\Gamma$ と $\rho$ の取り方によらないので、$\Gamma = \emptyset$、$\rho$ は空集合を定義域とする一意に定まる写像として構いません。この場合、$[\![P]\!] = [\![P]\!]_\rho$ は $\{0,1\}$ の要素と同一視できます。

たとえば、プログラム $(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})$ の表示的意味は、

$$\begin{array}{lcl}
[\![(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})]\!] & = & [\![(\mathsf{x\ and\ F})]\!]_{[x \mapsto [\![(\mathsf{not\ F})]\!]]} \\
& = & [\![(\mathsf{x\ and\ F})]\!]_{[x \mapsto (1 - [\![\mathsf{F}]\!])]} \\
& = & [\![(\mathsf{x\ and\ F})]\!]_{[x \mapsto (1-0)]} \\
& = & [\![(\mathsf{x\ and\ F})]\!]_{[x \mapsto 1]} \\
& = & [\![\mathsf{x}]\!]_{[x \mapsto 1]} \cdot [\![\mathsf{F}]\!]_{[x \mapsto 1]} \\
& = & 1 \cdot 0 \\
& = & 0
\end{array}$$

となります。前に見た操作的意味論と比較すると、計算結果こそ同じ（$P \Downarrow V$ ならば $[\![P]\!] = [\![V]\!]$）ですが、表示的意味論の計算は必ずしも操作的意味論が定める計算ステップとは対応していないことに注意してください。

**FB** の (共通する自由変数を持つ) プログラム上に、同値関係 $\sim$ を

$$P \sim P' \ \Leftrightarrow [\![P]\!] = [\![P']\!]$$

で定めます。閉じたプログラム $P, P'$ については、$P \sim P'$ であることと、$P$ と $P'$ の計算結果が一致すること（ある $V$ について $P \Downarrow V$ かつ $P' \Downarrow V$ となること）が一致します。

操作的意味論がプログラムの実際の挙動に基づくものであるのに対し、表示的意味論は、プログラムの抽象的な内容（**FB** なら真偽値および真偽値関数）に着目します。抽象化する際に、プログラムの細かい計算ステップなどの情報を気にする必要はありません。**FB** の場合は単純すぎてこれ以上どうしようもありませんが、より複雑な、現実的なプログラミング言語については、プログラムの持つ情報の、特に知りたい部分だけをなんとかして上手に取りだすことが大変重要になってきますが、そこが表示的意味論の醍醐味ともいうべきところです。プログラムの性質のうち、プログラムの正しさの検証や改良に役立てるために必要となる部分を抽象化して取り出す、というわけです。さらに、抽象化することで、あるプログラミング言語について得られた結果を、類似した構造を持つ多くのプログラミング言語にも共通して利用できるようになる、汎用性・一般性も表示的意味論の長所です。それに対し、操作的意味論は、具体的であるがゆえに、すこし規則を変更するだけでプログラムの意味が大きく変わってしまうかも知れないという精密さ・繊細さを持っています。現在のプログラム意味論は、操作的意味論と表示的意味論を密接に関連させ、両者の長所を生かして研究・応用されています。

## 1.4 表示的意味論＝計算の不変量

先に、表示的意味論が、プログラムの抽象的な内容に着目するものであると述べました。その内容の決め方は実はかなり自由なのですが、最低限外せないポイントとして、表示的意味論は計算の実行の過程において不変であるように定める、ということがあります。たとえば、**FB** の操作的意味論が定める計算の過程

$$(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)}) \to (\mathsf{let\,x\,be\,T\,in\,(x\,and\,F)}) \to (\mathsf{T\,and\,F}) \to \mathsf{F}$$

を見ると、

$$[\![(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})]\!] = [\![(\mathsf{let\,x\,be\,T\,in\,(x\,and\,F)})]\!] = [\![(\mathsf{T\,and\,F})]\!] = [\![\mathsf{F}]\!]$$

と、表示的意味は各計算ステップの前後で変化していないことがわかります。

クラインは、有名なエルランゲン・プログラムの中で、「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」と述べ、幾何学を、図形の変換の不変量を研究する数学として特徴づけました。プログラミング言語の意味論についても、クライン流の見方がとても有効であると、私は考えています。うんと乱暴にまとめるなら、プログラムの変換＝計算過程を定めるのが操作的意味論であり、一方、プログラムの計算の過程で不変な性質＝不変量を調べるのが表示的意味論の役割と言えます。

この、表示的意味論＝計算の不変量という視点は、プログラム意味論と数学（特に幾何学）を概念的に対比させ理解するうえで大変便利ですが、実は、技術的にも、プログラム意味論で用いられる圏論的な構造が、低次元トポロジー、特に結び目の不変量において用いられる圏論的な構造と密接に関連していることがわかっています。このことについては、講義の後のほうで触れたいと思います。

## 1.5 この講義の構成

通常のプログラミング言語の理論の講義では、考察の対象となるプログラミング言語の構文・文法の定義をきちんと与え、然る後に意味論等の議論をはじめます。実際、**FB** については、おおむねそ

の定跡を踏襲して解説しました。しかし、この講義では、今後は、プログラムの構文論はほとんど扱いません。最大の理由は、構文論の説明は、時間と手間が非常にかかるということ、そして、（構文論の専門家からはお叱りを受けると思いますが）構文論の大半は教えるのも教わるのも退屈だということです。限られた時間のなかで圏論を用いたプログラム意味論の本質（と私が思うこと）をお伝えするために、構文論はほぼ無視します。

もうひとつ、以下の講義では、基本的に表示的意味論だけを扱い、操作的意味論に触れることはあまりありません。これは、構文論を無視するために、構文論に深く依存する操作的意味論について論じることが難しい、ということもありますが、なにより、圏論に基づく意味論は、表示的意味論のほうがより基本的であり、操作的意味論について圏論的に論じるためには、表示的意味論を十分に理解することが不可避である、という理由があります。

また、圏論的なプログラム意味論（表示的意味論）の全部について語ることはもとより無理ですので、思い切り題材を絞り込み、以下のような進め方をしたいと思います。まず、圏論的な意味論の一般的な方針について説明します。然る後、ふたつの重要なケースについて紹介します。ひとつは、カルテジアン閉圏と呼ばれる構造に関する話題です。これは、ラムダ計算と呼ばれる計算モデルや、関数型プログラミング言語、さらに定理証明支援システムなどの意味論の、もっとも基礎的な部分になります。

もうひとつは、トレース付きモノイダル圏に関する話題です。こちらは、再帰プログラムの意味論や、相互作用の幾何と呼ばれる計算モデルの基礎となるものですが、実は結び目の量子不変量とも深い関係があります。

他にも重要なケースは多々ありますが、（１）分野の研究者の間で重要性が広く認められていること、（２）それほど説明がむつかしくないこと、そして（３）応用が豊富なこと、（４）講義の流れが自然になること、という観点からこれらに注目することにしました。（もちろん、私にとってなじみ深い話題であるということも大きな理由です。）

この話題の性質上、圏論の知識は全く仮定しないというわけにはいきません。基礎的な圏論については、講義中にもなるべく補足していきたいと思いますが、最小限、これだけ知っていればたぶん大丈夫、というまとめを末尾（付録 A）につけておきますので、参考にしていただけますと幸いです。

## 2　圏論的な意味論

### 2.1　FB の意味論再考

**FB** では、プログラム $P$ の表示的意味は、真偽値関数

$$[\![P]\!] : \{0,1\}^{fv(P)} \longrightarrow \{0,1\}$$

として定められていました。実は、集合 $\{0,1\}$ は任意の集合 $X$ に置き換えても問題ありません（$[\![\mathsf{T}]\!]$ と $[\![\mathsf{F}]\!]$ を区別するためには、ふたつ以上の要素を持つ集合である必要はあります）。また、自由変数の集合 $fv(P)$ は有限集合ですので、プログラム $P$ の表示的意味は、写像

$$[\![P]\!] : X^n \longrightarrow X$$

（ただし $n = |fv(P)|$）により与えられます。

さらに、ここまで集合間の写像としてきましたが、実のところ、$[\![P]\!]$ を、適当な圏の適当な対象 $X$ について、$X^n$ から $X$ への射、としてしまっても何ら差し支えありません。ただし、どんな圏でも良いというわけではなく、有限直積を持つことが必要です。

## 2.2 有限直積を持つ圏

圏 $\mathcal{C}$ の有限個の対象 $X_1, \cdots, X_n$ の**直積**は、対象 $P$ と射 $\pi_i : P \to X_i$ $(i = 1, \cdots, n)$（**射影**と呼ばれます）の組で、以下の条件を満たすものです。

任意の対象 $Z$ と射 $f_i : Z \to X_i$ $(i = 1, \cdots, n)$ に対し、

$$f_i = \pi_i \circ h \qquad (i = 1, \cdots, n)$$

となる射 $h : Z \to P$ がただひとつ存在する。

対象 $X_1, \cdots, X_n$ の直積は、存在すれば、同型を除いて一意に定まります（複数あっても、それらはすべて互いに同型になります）。以下では、圏 $\mathcal{C}$ はすべての有限直積を持つものとし、対象 $X_1, \cdots, X_n$ の直積を (同型なもののうちから) ひとつ固定して対象 $X_1 \times \cdots \times X_n$ および射 $\pi_i : X_1 \times \cdots \times X_n \to X_i$ $(i = 1, \cdots, n)$ の組で表すことにします。そして、対象 $Z$ と射 $f_i : Z \to X_i$ $(i = 1, \cdots, n)$ に対し、$f_i = \pi_i \circ h$ $(i = 1, \cdots, n)$ となる射 $h$ を、

$$\langle f_1, \cdots, f_n \rangle : Z \to X_1 \times \cdots \times X_n$$

で表すことにします。すると、

$$\pi_i \circ \langle f_1, \cdots, f_n \rangle = f_i \quad \langle \pi_1 \circ g, \cdots, \pi_n \circ g \rangle = g$$

(ただし $g : Z \to X_1 \times \cdots \times X_n$) が成り立ちます。

### 諸定義

- $n = 0$ の場合の直積を対象 1 で表すと、任意の対象 $Z$ に対し、$Z$ から 1 への射 $\langle\,\rangle : Z \to 1$ が唯一つ存在することになります。このような対象は終対象と呼ばれます。
- 射 $f_i : X_i \to Y_i$ $(i = 1, 2, \ldots, n)$ に対し、射 $f_1 \times \ldots \times f_n : X_1 \times \ldots \times X_n \to Y_1 \times \ldots \times Y_n$ を、$f_1 \times \ldots \times f_n = \langle f_1 \circ \pi_1, \ldots, f_n \circ \pi_n \rangle$ で定めます。この定義により、有限直積を持つ圏 $\mathcal{C}$ において、写像 $(X_1, \ldots, X_n) \mapsto X_1 \times \ldots \times X_n : (Ob(\mathcal{C}))^n \to Ob(\mathcal{C})$ は $\mathcal{C}^n$ から $\mathcal{C}$ への関手に拡張されます。

## 2.3 FB の圏論的意味論

有限直積を持つ圏で **FB** の意味論を与えてみましょう。$\mathcal{C}$ を有限直積を持つ圏とし、$X$ を $\mathcal{C}$ の対象とします。また、射 $tt, ff : 1 \to X$, $not : X \to X$ および $and : X^2 \to X$ が存在し、

$$\begin{array}{rcl@{\qquad}rcl} not \circ tt & = & ff & not \circ ff & = & tt \\ and \circ \langle tt, tt \rangle & = & tt & and \circ \langle tt, ff \rangle & = & ff \\ and \circ \langle ff, tt \rangle & = & ff & and \circ \langle ff, ff \rangle & = & ff \end{array}$$

を満たすものとします。

**FB** のプログラム $P$ と、互いに異なる変数の列 $x_1, \ldots, x_n$ について、$fv(P)$ の要素が $x_1, \ldots, x_n$ の中に含まれているとき、$[\![x_1, \ldots, x_n \vdash P]\!] : X^n \to X$ を以下のように定めます。

$$\begin{array}{ll} [\![x_1, \ldots, x_n \vdash x_i]\!] & = \pi_i \\ [\![x_1, \ldots, x_n \vdash \mathsf{T}]\!] & = tt \circ \langle\,\rangle \\ [\![x_1, \ldots, x_n \vdash \mathsf{F}]\!] & = ff \circ \langle\,\rangle \\ [\![x_1, \ldots, x_n \vdash (\mathsf{not}\ P)]\!] & = not \circ [\![x_1, \ldots, x_n \vdash P]\!] \\ [\![x_1, \ldots, x_n \vdash (P_1\ \mathsf{and}\ P_2)]\!] & = and \circ \langle [\![x_1, \ldots, x_n \vdash P_1]\!], [\![x_1, \ldots, x_n \vdash P_2]\!] \rangle \\ [\![x_1, \ldots, x_n \vdash (\mathsf{let}\ x\ \mathsf{be}\ P_1\ \mathsf{in}\ P_2)]\!] & = \\ \qquad & [\![x_1, \ldots, x_n, x \vdash P_2]\!] \circ \langle \pi_1, \ldots, \pi_n, [\![x_1, \ldots, x_n \vdash P_1]\!] \rangle \end{array}$$

たとえば、プログラム $(\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})$ の $\mathcal{C}$ における意味は、以下のように求められます。

$$\begin{aligned}
[\![\vdash (\mathsf{let\,x\,be\,(not\,F)\,in\,(x\,and\,F)})]\!] &= [\![\mathsf{x} \vdash (\mathsf{x\,and\,F})]\!] \circ [\![\vdash (\mathsf{not\,F})]\!] \\
&= \mathit{and} \circ \langle [\![\mathsf{x} \vdash \mathsf{x}]\!], [\![\mathsf{x} \vdash \mathsf{F}]\!] \rangle \circ \mathit{not} \circ [\![\vdash \mathsf{F}]\!] \\
&= \mathit{and} \circ \langle \mathit{id}, \mathit{ff} \circ \langle\,\rangle \rangle \circ \mathit{not} \circ \mathit{ff} \\
&= \mathit{and} \circ \langle \mathit{id}, \mathit{ff} \circ \langle\,\rangle \rangle \circ \mathit{tt} \\
&= \mathit{and} \circ \langle \mathit{tt}, \mathit{ff} \circ \langle\,\rangle \circ \mathit{tt} \rangle \\
&= \mathit{and} \circ \langle \mathit{tt}, \mathit{ff} \rangle \\
&= \mathit{ff}
\end{aligned}$$

特に、$\mathcal{C}$ を **Set**、$X$ を $\{0,1\}$ とし、$\mathit{tt}$、$\mathit{ff}$、$\mathit{not}$、$\mathit{and}$ を適切に定めれば、前に与えた **FB** の表示意味論は、この圏論的な意味論の一例となっていることがわかります（$[\![P]\!]_\rho = [\![x_1, \ldots, x_n \vdash P]\!] \circ \rho$ が成り立ちます）。

**直積を持つ圏の例と表示的意味論**　表示的意味論でよく用いられる圏のひとつに、完備半順序集合（正確には最小元を持つ $\omega$-完備半順序集合）の圏 **Cpo** があります。半順序集合 $(X, \sqsubseteq)$ が最小の要素を持ち、$X$ の任意の可算単調列

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

が上限 $\bigsqcup x_i$ を持つとき、$(X, \sqsubseteq)$ は完備半順序集合と呼ばれます。完備半順序集合 $(X, \sqsubseteq)$ の最小元を、しばしば $\bot_X$ と表記します。完備半順序集合 $(X, \sqsubseteq)$ と $(Y, \sqsubseteq)$ について、写像 $f : X \to Y$ 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います。**Cpo** は、完備半順序集合を対象とし、連続写像を射に持つ圏であり、有限直積を持ちます。直感的には、完備半順序集合の最小元は、値が未定義な計算＝停止しない計算をあらわします。実際的なプログラミング言語では、とまらないプログラムを書けることも実はとても重要です。特に、再帰プログラム（自分自身を呼び出すように書かれたプログラム）を持つ言語では、非停止性は避けては通れませんが、**Cpo** を用いて、そのような再帰プログラムの意味論を非常にエレガントにあたえることが可能です。この話題については、また後程触れます。

$\{0,1\}_\bot$ を、要素を $0, 1, \bot$ とし、順序が $x \sqsubseteq y \;\Leftrightarrow\; (x = \bot_X$ または $x = y)$ で定まる完備半順序集合とします。**FB** の圏論的意味論で、$\mathcal{C}$ を **Cpo**、$X$ を $\{0,1\}_\bot$ とし、$\mathit{tt}$、$\mathit{ff}$、$\mathit{not}$、$\mathit{and}$ を適切に定めれば、**Cpo** における **FB** の表示的意味論が得られます。**FB** の計算はすべて停止するので、**Cpo** を使うご利益は **FB** そのものに対してはありませんが、**FB** を再帰プログラムを許すように拡張すると、**Cpo** を用いた意味論が役に立ちます。

**有限直積を保存する関手**　有限直積を持つ圏 $\mathcal{C}$ および $\mathcal{D}$ について、関手 $F : \mathcal{C} \to \mathcal{D}$ が有限直積を保存するとは、$\mathcal{C}$ の対象 $X_1, \ldots, X_n$ の直積 $X_1 \times \ldots \times X_n$ とその射影 $\pi_i : X_1 \times \ldots \times X_n \to X_i$ $(i = 1, \ldots, n)$ に対し、$F(X_1 \times \ldots \times X_n)$ と $F(\pi_i) : F(X_1 \times \ldots \times X_n) \to F(X_i)$ $(i = 1, \ldots, n)$ が、$\mathcal{D}$ における $F(X_1), \ldots, F(X_n)$ の直積になることをいいます。

今、圏 $\mathcal{C}(\mathbf{FB})$ を以下のように定めます：

- $\mathcal{C}(\mathbf{FB})$ の対象は非負整数 $0, 1, 2, \ldots$
- $\mathcal{C}(\mathbf{FB})$ における $m$ から $n$ への射は、自由変数が $x_1, \ldots, x_m$ に含まれているようなプログラム $P_1 \ldots P_n$ の $\sim$ の同値類の組 $([P_1], \ldots, [P_n])_{(x_1, \ldots, x_m)}$

図 1: 様々なプログラミング言語の機能と対応する圏論的構造の例

| | |
|---|---|
| 一階関数型プログラム | 有限直積を持つ圏 |
| 高階関数型プログラム | カルテジアン閉圏 |
| 副作用を伴うプログラム | 有限直積を持つ圏上の強モナド |
| 再帰プログラム | 有限直積を持つ圏上の不動点演算子 |
| 依存型 | 局所カルテジアン閉圏 |
| 線形型 | モノイダル圏 |
| 相互作用の幾何 | トレース付きモノイダル圏 |
| 種々のデータ構造 | 関手の始代数、終代数、双代数 |

- $n$ 上の恒等射は $([x_1], \ldots, [x_n])_{(x_1,\ldots,x_n)}$
- $([P_1], \ldots, [P_m])_{(x_1,\ldots,x_l)} : l \to m$ と $([Q_1], \ldots, [Q_n])_{(x_1,\ldots,x_m)} : m \to n$ の合成は

$$([\mathsf{let}\, x_1 \,\mathsf{be}\, P_1 \,\mathsf{in}\, \ldots \mathsf{let}\, x_m \,\mathsf{be}\, P_m \,\mathsf{in}\, Q_1], \ldots, [\mathsf{let}\, x_1 \,\mathsf{be}\, P_1 \,\mathsf{in}\, \ldots \mathsf{let}\, x_m \,\mathsf{be}\, P_m \,\mathsf{in}\, Q_n])_{(x_1,\ldots,x_l)}$$

$\mathcal{C}(\mathbf{FB})$ は有限直積を持つ圏です（$m_1, \ldots, m_n$ の直積は $m_1 + \ldots + m_n$ です）。実は、有限直積を持つ圏 $\mathcal{C}$ に対し、$\mathcal{C}(\mathbf{FB})$ から $\mathcal{C}$ への直積を保存する関手を与えることは、$\mathcal{C}$ における $\mathbf{FB}$ の意味論を与えることと、ぴったり対応しています。どういうことかというと、$\mathcal{C}(\mathbf{FB})$ の対象 1 を関手でうつした先を $X$ とし、$and$ を $([x_1 \,\mathsf{and}\, x_2]_{(x_1,x_2)}) : 2 \to 1$ を関手で写した先、などとすれば、$\mathcal{C}$ における $\mathbf{FB}$ の意味論を得ることができますし、逆に、$\mathcal{C}$ における $\mathbf{FB}$ の意味論から、$\mathcal{C}(\mathbf{FB})$ から $\mathcal{C}$ への直積を保存する関手を構成することも可能です。

ここまで $\mathbf{FB}$ の意味論について細かく見てきましたが、圏論の言葉で要約すると、$\mathbf{FB}$ の表示的意味論を与えることは、$\mathcal{C}(\mathbf{FB})$ からの直積を保存する関手を与えることに他なりません。

### 2.4 圏論的意味論＝構造を保存する関手

このように、$\mathbf{FB}$ のような単純な関数型プログラミング言語については、有限直積を持つ圏と、有限直積を保存する関手が表示的意味論を与えます。これは偶然ではなく、より複雑な言語についても、同様のことが言えます。ただし、言語の機能に応じて、直積だけでなく、さまざまな構造を持つ圏と、それらの構造を保存する関手を考える必要があります。圏論を用いたプログラム意味論では、そのような構造を特定し、構造を持つ圏・構造を保存する関手の性質や具体例を調べることが本質的です。図 1 に、よく用いられる構造の例を列挙してみました。

この講義の残りでは、そのような構造の代表例として、カルテジアン閉圏とトレース付きモノイダル圏について解説します。

## 3 カルテジアン閉圏

### 3.1 型付きラムダ計算と関数型プログラミング

多くのプログラミング言語では、プログラムをデータとみなすことができます。つまり、プログラムをデータとして受け取り、処理するプログラムを書くことが可能です。さらに、「プログラムを処理するプログラム」をまたデータとして受け取るようなプログラムも考えることができます。一般に、型付きラムダ計算や（高階）関数型プログラミング言語では、データ型 $\sigma_1$ と $\sigma_2$ に対し、$\sigma_1$ のデー

タを入力とし $\sigma_2$ のデータを出力するプログラムのデータ型（関数型）$\sigma_1 \to \sigma_2$ が用意されています。たとえば、整数のデータ型を $\mathsf{int}$ であらわすと、整数上の関数を入力とし、整数を出力する（汎）関数の型は $(\mathsf{int} \to \mathsf{int}) \to \mathsf{int}$ となります。

このような関数型の意味論を与えるのが、カルテジアン閉圏です。

### 3.2 カルテジアン閉圏

有限直積を持つ圏 $\mathcal{C}$ の対象 $X$ と $Y$ に対し、$X$ と $Y$ の冪とは、対象 $E$ と射 $\varepsilon : E \times X \to Y$ の組で、以下の条件を満たすものです。

> 任意の対象 $Z$ と射 $f : Z \times X \to Y$ に対し、
>
> $$\varepsilon \circ (h \times id_X) = f$$
>
> となる射 $h : Z \to E$ がただ一つ存在する。

以下では、そのような対象 $E$ をひとつ選んで $Y^X$ と表記し、$f : Z \times X \to Y$ に対し一意に定まる $h : Z \to Y^X$ を $\overline{f}$ で表すことにします。すると、等式

$$\begin{aligned} \varepsilon \circ (\overline{f} \times id_X) &= f \quad (f : Z \times X \to Y) \\ \overline{\varepsilon \circ (h \times id_X)} &= h \quad (h : Z \to Y^X) \end{aligned}$$

が成り立ちます。

有限直積とすべての冪をもつ圏を、**カルテジアン閉圏**と呼びます。

例

- 集合と写像の圏 $\mathbf{Set}$ は、カルテジアン閉圏です。集合 $X$ と $Y$ の冪は、冪集合 $Y^X = \{f : X \to Y\}$ および $\varepsilon(f, x) = f(x)$ で定められる写像 $\varepsilon : Y^X \times X \to Y$ によって与えられます。$f : Z \times X \to Y$ に対し、$\overline{f} : Z \to Y^X$ は、$(\overline{f}(z))(x) = f(z, x)$ で定まります。
- 任意の圏 $\mathcal{C}$ に対し、関手の圏 $[\mathcal{C}, \mathbf{Set}]$ （しばしば前層の圏と呼ばれます）はカルテジアン閉圏です。
- 完備半順序集合と連続写像の圏 $\mathbf{Cpo}$ はカルテジアン閉圏です。完備半順序集合 $X = (X, \sqsubseteq)$ と $Y = (Y, \sqsubseteq)$ に対し、冪対象は、$X$ から $Y$ への連続写像全体の集合 $Y^X$ に、半順序

  $$f \sqsubseteq g \;\Leftrightarrow\; \text{すべての } x \in X \text{ について } f(x) \sqsubseteq g(x)$$

  を入れたものと、$\varepsilon(f, x) = f(x)$ で定められる写像 $\varepsilon : Y^X \times X \to Y$ によって与えられます。
- ベクトル空間の圏 $\mathbf{Vect}_K$ と有限次元ベクトル空間の圏 $\mathbf{Vect}_K^{\mathbf{fd}}$ は、いずれも有限直積を持ちます（直和空間で与えられます）が、これらの圏は冪を持たないので、カルテジアン閉圏ではありません。

**単純型付きラムダ計算の意味論**　ここでは詳細には立ち入りませんが、**単純型付きラムダ計算** $\lambda_\to$ と呼ばれる計算系があります。$\lambda_\to$ は、基底型と呼ばれる基礎的な型と、関数型だけが用意されている、すべての高階関数型プログラミング言語の雛形ともいうべきものです。$\mathbf{FB}$ から有限直積を持つ圏 $\mathcal{C}(\mathbf{FB})$ を構成したのと全く同じ流儀で、$\lambda_\to$ から、カルテジアン閉圏 $\mathcal{C}(\lambda_\to)$ を構成することができます。そして、$\mathbf{FB}$ の場合と同様に、単純型付きラムダ計算のカルテジアン閉圏 $\mathcal{C}$ における表示的意味論を、$\mathcal{C}(\lambda_\to)$ から $\mathcal{C}$ へのカルテジアン閉圏の構造を保存する関手として定式化することができます。

# 4　トレース付きモノイダル圏

ここからは、少々駆け足になりますが、テンソル積やトレースを持つ圏と、そのプログラム意味論における応用について紹介します。

## 4.1　再帰プログラムの不動点意味論

プログラム意味論における古典的な話題である、**再帰プログラム**の表示的意味論について、簡単な例をもとに説明します。 整数を入力に取り整数を出力する簡単なプログラム

$$\mathsf{sum}(x) \ \equiv \ \mathsf{if}\ x = 0\ \mathsf{then}\ 0\ \mathsf{else}\ x + \mathsf{sum}(x-1)$$

を考えます。これは、プログラムの定義（$\equiv$ の右辺）で自身（sum）を呼び出す再帰プログラムになっています。たとえば、$\mathsf{sum}(3)$ の実行は、$3 \neq 0$ なので $3 + \mathsf{sum}(2)$ と sum 自身を呼び出し、以下同様に続けて $3 + (2 + (1 + \mathsf{sum}(0)))$ に到達し、$\mathsf{sum}(0) = 0$ なので和 $3 + (2 + (1 + 0))$ を計算して 6 を答えとして返します。sum のような再帰プログラムは、その実行が常に停止するとは限らないので、整数上の部分関数を定めます。そして、sum の定める部分関数は、部分関数全体の上の（汎）関数

$$F(f)(x) \ = \ \begin{cases} 0 & (x{=}0) \\ x{+}f(x{-}1) & (x{\neq}0 \text{ かつ } f(x-1) \text{ が定義されている場合}) \\ \text{未定義} & (\text{その他の場合}) \end{cases}$$

の不動点として理解できます。sum に限らず、再帰プログラムの表示的意味は、適切な写像の不動点として与えることができます。圏論を用いた表示的意味論では、そのような再帰プログラムの意味論に必要な不動点を取る操作は、以下のように定式化されます。有限直積を持つ圏 $\mathcal{C}$ において、以下の条件をみたす写像の族 $(-)^\dagger : \mathcal{C}(A \times X, X) \to \mathcal{C}(A, X)$ を**コンウェイ不動点演算子**と呼びます：

- $f : A \times X \to X$, $h : A' \to A$ について $f^\dagger \circ h = (f \circ (h \times id_X))^\dagger : A' \to X$
- $f : A \times Y \to X$, $g : A \times X \to Y$ について $(f \circ \langle \pi_{A,X}, g \rangle)^\dagger = f \circ \langle id_A, (g \circ \langle \pi_{A,Y}, f \rangle)^\dagger \rangle : A \to X$
- $f : A \times X \times X \to X$ について $f^{\dagger\dagger} = (f \circ (id_A \times \Delta_X))^\dagger : A \to X$

これらの条件は、再帰プログラムの不動点意味論のほとんどすべてについて成立しています。代表的な例として、完備半順序集合の圏 **Cpo** があげられます。**Cpo** では、連続写像 $f : (X, \sqsubseteq) \to (X, \sqsubseteq)$ は最小の不動点を持ち、それは可算単調列

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

の上限 $\bigsqcup f^n(\bot)$ で与えられます。先の sum を例にとると、汎関数 $F$ は整数上の部分関数全体がなす完備半順序集合の上の連続写像であり、その最小不動点は sum の定める部分関数に他なりません。

以上で述べた圏論的な不動点意味論は、おおむね１９８０年代までに完成された、標準的なものです。１９９０年代になって、トレース付きモノイダル圏という構造が不動点意味論と関係づけられてから、この分野の見方に変化が生じました。この講義の残りでは、その比較的新しい展開について、時間の許す範囲で紹介します。

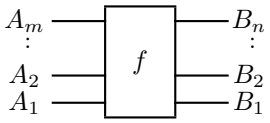
## 4.2　モノイダル圏とトレース

**モノイダル圏**　**モノイダル圏**あるいは**テンソル圏** $\mathcal{C} = (\mathcal{C}, \otimes, I, a, l, r)$ とは、圏 $\mathcal{C}$ と、それに付随する**テンソル積**または**モノイダル積**と呼ばれる関手 $\otimes : \mathcal{C}^2 \to \mathcal{C}$、単位対象と呼ばれる $\mathcal{C}$ の対象 $I$、テンソ

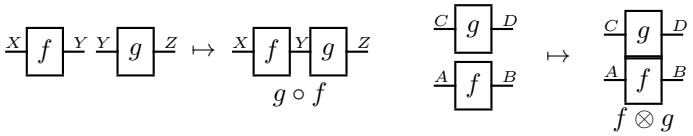
ル積と単位対象の結合律・単位律に相当し適当な公理をみたす可逆な自然変換 $a_{A,B,C} : (A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C)$, $l_A : I \otimes A \xrightarrow{\sim} A$ および $r_A : A \otimes I \xrightarrow{\sim} A$ の組のことをいいます。$a, l, r$ がすべて恒等射であるようなモノイダル圏は、厳格であるといいます。厳格なモノイダル圏では、$A \otimes (B \otimes C)$ と $(A \otimes B) \otimes C$ は同一の対象であり、$A \otimes I$、$I \otimes A$ と $A$ についても同様です。任意のモノイダル圏は、厳格なモノイダル圏と（モノイダル圏の構造を厳格に保存する関手を介して）同値です（コヒーレンス定理）。ですから、モノイダル圏について考察する際には、$a, l, r$ のことは忘れて、あたかも厳格なモノイダル圏を扱っているようにみなしても問題ありません。以下でも、$a, l, r$ のことは無視して話を進めます。

**ブレイド、バランス、対称性**　モノイダル圏で、条件 $c_{A,B\otimes C} = (id_B \otimes c_{A,C}) \circ (c_{A,B} \otimes id_C)$ および $c^{-1}_{B\otimes C,A} = (id_B \otimes c^{-1}_{C,A}) \circ (c^{-1}_{B,A} \otimes id_C)$ を満たす可逆な自然変換 $c_{A,B} : A \otimes B \xrightarrow{\sim} B \otimes A$ を持つものを、**ブレイド付きモノイダル圏**と呼び、$c$ は**ブレイド**と呼ばれます。さらに、**バランス**あるいは**ツイスト**と呼ばれる可逆な自然変換 $\theta_A : A \xrightarrow{\sim} A$ が存在して $\theta_I = id_I$ および $\theta_{A\otimes B} = c_{B,A} \circ (\theta_B \otimes \theta_A) \circ c_{A,B}$ をみたすブレイド付きモノイダル圏を、**バランス付きモノイダル圏**と呼びます。バランス付きモノイダル圏で、$\theta = id$（したがって $c = c^{-1}$）が成り立つものは、**対称モノイダル圏**（symmetric monoidal category）と呼ばれます。

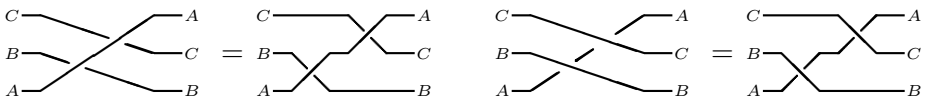
**ストリング図式**　モノイダル圏の射は、簡潔に図示することができ（しばしばペンローズ図式やストリング図式と呼ばれます）、それは図の連続的な変形にたいして整合的です。この講義では、射 $f : A_1 \otimes A_2 \otimes \ldots \otimes A_m \to B_1 \otimes B_2 \otimes \ldots \otimes B_n$ を



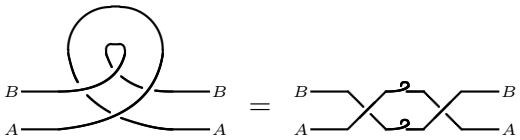

とあらわします。射の合成やテンソル積は、それぞれの射が対応する図の直列つなぎおよび並列つなぎであらわします。

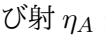


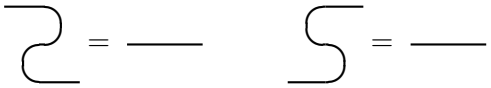

ブレイド $c_{A,B} : A \otimes B \to B \otimes A$ は交差  として、また $c^{-1}_{A,B} : B \otimes A \to A \otimes B$ は逆向きの交差  としてあらわします。ブレイドの条件式は、以下のように図示できます。



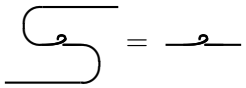

バランス $\theta_A : A \to A$ は、ねじれ  であらわします。$\theta_A^{-1}$ は逆向きのねじれ  です。バランスの条件式も以下のようにあらわせます。



**双対性とリボン圏**　次に、モノイダル圏における双対性の概念を導入します。モノイダル圏の対象 $A$ の（左）双対とは、対象 $A^*$ および射 $\eta_A : I \to A \otimes A^*$ と $\varepsilon_A : A^* \otimes A \to I$ （それぞれ  と  で図示します）で、条件



を満たすものです。**リボン圏**とは、すべての対象が双対を持ち、

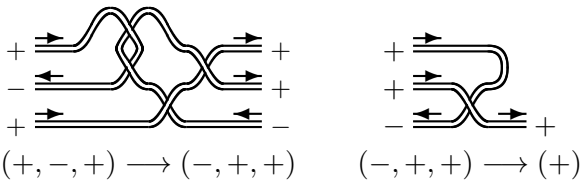


が成り立つようなバランス付きモノイダル圏です。なお、対称な（$\theta = 1, c = c^{-1}$ であるような）リボン圏は、しばしば**コンパクト閉圏**とも呼ばれます。

たとえば、体 $K$ について、$K$ 上の有限次元線形空間を対象に、また線形写像を射に持つ圏 $\mathbf{Vect}_K^{\mathbf{fd}}$ は、対称なリボン圏（すなわちコンパクト閉圏）です。この場合、テンソル積 $U \otimes V$ は線形空間のテンソル積 $U \otimes_K V$、単位対象は $K$、ブレイド $c_{U,V}$ は $U \otimes V$ から $V \otimes U$ への自明な同型写像、バランスは恒等写像であり、$U$ の双対 $U^*$ は双対空間 $\hom(U, K)$ により与えられます。

似た例では、対象が集合、射が集合の間の二項関係である圏 **Rel** があります。**Rel** の恒等射 $id_X : X \to X$ は恒等関係 $\{(x, x) \mid x \in X\}$、また二項関係 $R : X \to Y$ と $S : Y \to Z$ の合成 $S \circ R : X \to Z$ は $\{(x, z) \in X \times Z \mid \exists y \in Y\ (x, y) \in R, (y, z) \in R\}$ で与えられます。テンソル積 $X \otimes Y$ は集合の直積 $X \times Y$、単位対象は一点集合 $1$ です。集合 $X$ の双対は $X$ 自身です。**Rel** は、非決定的な計算のモデルとして、また線形論理のモデルとして、プログラム意味論でよく用いられている圏です。

**量子不変量とリボン圏**　結び目の理論では、以下のような（向きづけられた、枠付きの）タングルの圏 **Tangle** を考えます。**Tangle** の対象は、（タングルの向きを表わす）記号 $+$ と $-$ の有限列です。そして、$(s_1, \ldots, s_m)$ から $(s'_1, \ldots, s'_n)$ への射は、下図のような、$+$ では左から右に、$-$ では右から左に向き付けられた、$s_1, \ldots, s_m, s'_1, \ldots, s'_n$ を端点とするタングルの、連続変形に関する同値類です。



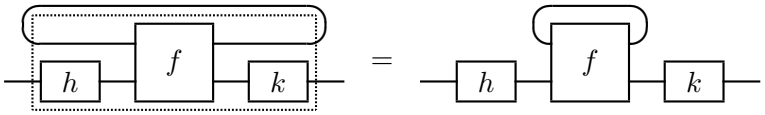

**Tangle** は、リボン圏です。それも、ただのリボン圏ではなく、ひとつの対象から自由生成されたリボン圏です。したがって、任意のリボン圏 $\mathcal{C}$ とその対象をひとつ選ぶことにより、**Tangle** から $\mathcal{C}$ へのリボン圏の構造を保存する関手が定まり、この関手からタングルの不変量がひとつ得られます。たとえば、**量子群**から、その表現の圏としてリボン圏を構成することができます。すなわち、量子群から、タングルの不変量を構成することができるわけです。

同様のことは、集合と二項関係の圏 **Rel** についても可能です。**Rel** の中で量子群に相当するもの（リボンホップ代数）を見つけ、その” 表現” の圏として非対称なリボン圏を得ることができます。

**トレース付きモノイダル圏**　トレース付きモノイダル圏には、二つの特徴付けがあります。一つは、抽象的なトレース演算子を備えたバランス付きモノイダル圏というもので、これが Joyal, Street, Verity の原論文における定義でもあります。もう一つは、同じ論文で与えられている構造定理によるもので、トレース付きモノイダル圏は、必ずあるリボン圏の充満部分モノイダル圏になっている、というものです。トレース付きモノイダル圏では、対象の双対は必ずしも存在しません。構造定理については後で触れることにして、まず、トレース演算子による特徴付けから説明します。

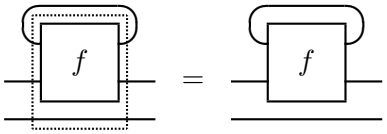
図 2: トレースの公理

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

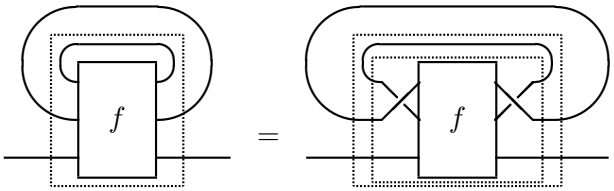



$$Tr^X_{X,X}(c_{X,X}) \circ \theta^{-1}_X = id_X = Tr^X_{X,X}(c^{-1}_{X,X}) \circ \theta_X$$

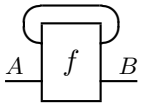

$$Tr^X_{C\otimes A,C\otimes B}(id_C \otimes f) = id_C \otimes Tr^X_{A,B}(f)$$




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




バランス付きモノイダル圏 $\mathcal{C}$ が与えられているものとします。このとき、$\mathcal{C}$ 上のトレース演算子とは、$\mathcal{C}$ の対象で添字付けられた写像の族 $Tr^X_{A,B} : \mathcal{C}(A \otimes X, B \otimes X) \to \mathcal{C}(A, B)$ で、図 2 の四つの条件を満たすものです。ここで、$f : A \otimes X \to B \otimes X$ のトレース $Tr^X_{A,B}(f) : A \to B$ を




で図示することにします。トレース演算子を持つバランス付きモノイダル圏を、**トレース付きモノイダル圏**と呼びます。

トレース付きモノイダル圏の例として、第一にリボン圏が挙げられます。任意のリボン圏はトレース演算子をただひとつ持ちます。有限次元線形空間の対称リボン圏 $\mathbf{Vect}^{\mathbf{fd}}_K$ では、トレース演算子は通常のトレース（行列の対角和）を対象 $A, B$ でパラメータ化したものです。通常のトレースは $A = B = I$ の場合、つまり $Tr^X_{I,I} : \mathcal{C}(X, X) \to \mathcal{C}(I, I)$ に相当します。量子群の表現のなすリボン圏では、トレース演算子は、量子トレースと呼ばれるもの（のパラメータ化）になります。

**Rel** では、二項関係 $R : A \times X \to B \times X$ のトレース $Tr^X_{A,B}R : A \to B$ は、$\{(a, b) \in A \times B \mid \exists x((a, x), (b, x)) \in R\}$ です。

リボン圏でないトレース付きモノイダル圏の例としては、**Tangle** の対象を $+$ の列だけに制限した充満部分圏 $\mathbf{Tangle}_+$ があります。（$\mathbf{Tangle}_+$ は、ひとつの対象から自由生成されたトレース付きモノイダル圏です）。**Rel** も別のテンソル積（有限双直積）についてトレース演算子を持つことが知られており、こちらはリボン圏ではありません。（このトレースは後で述べる相互作用の幾何でよく用

いられます。）一般に、リボン圏の充満部分モノイダル圏はトレース付きモノイダル圏ですが、後述する構造定理より、実はその逆も成り立ちます。

### 4.3　トレースと再帰プログラムの不動点意味論

実は、コンウェイ不動点演算子は、トレース演算子の特別な場合であるとみなすことができます。有限直積を持つ圏は、直積をテンソル積とみなすことにより、対称モノイダル圏の例になっています。有限直積をテンソル積とみなした対称モノイダル圏においては、コンウェイ不動点演算子とトレース演算子の間には一対一対応がつくことが、1990 年代中頃に、筆者と Hyland によって独立に示されました。このことから、**Cpo** など、プログラム意味論で用いられている多くの圏が、実はトレース付きモノイダル圏になっていることがわかります。結び目の理論で必要とされる非対称なリボン圏ではありませんが、対称リボン圏の断片が、プログラム意味論において（誰にも気づかれることなく）古くから用いられていた、というのは興味深いことだと思います。

この結果はさらに一般化でき、直積でないテンソル積を持つモノイダル圏においても、適当な（プログラム意味論でよく用いられるような）条件を満たすものにおいては、トレースから不動点演算子を構成することがわかっています。それを用いて、古典的なプログラム意味論では説明できなかった、巡回的な共有データ構造から生み出される再帰計算の意味論を展開することが可能です。

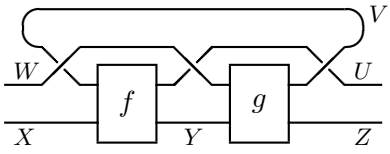
### 4.4　相互作用の幾何

トレース付きモノイダル圏のもうひとつの特徴付けのもととなる構造定理は、以下のようなものです。

> 任意のトレース付きモノイダル圏 $\mathcal{C}$ について、$\mathcal{C}$ からのトレース付きモノイダル圏の構造を保つ充満忠実関手があるようなリボン圏 $\mathbf{Int}\,\mathcal{C}$ が存在します。

ここで登場する **Int** 構成は、計算機科学において思いがけない応用をもたらしました。それは、**Int** 構成により得られるリボン圏が、双方向の情報のやりとりを伴なう計算（相互作用）の意味論を与えるために非常に便利なものだからです。

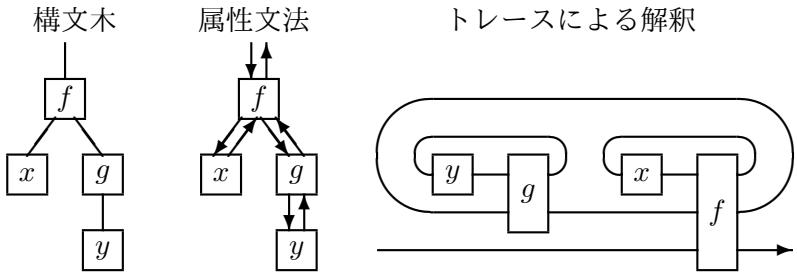
トレース付きモノイダル圏 $\mathcal{C}$ に対し、リボン圏 $\mathbf{Int}\,\mathcal{C}$ は、$\mathcal{C}$ の対象の組を対象とし、$\mathbf{Int}\,\mathcal{C}$ における $(X,U)$ から $(Y,V)$ への射は、$\mathcal{C}$ における $X\otimes V$ から $Y\otimes U$ への射として与えられます。射の定義の中で、対象の第 2 成分の順番がひっくり返っている（反変になっている）ことに注意してください。$\mathbf{Int}\,\mathbf{C}$ における射 $f:(X,U)\to(Y,V)$ と $g:(Y,V)\to(Z,W)$ の合成は、以下のように行います。




左から右への素直な流れと、トレースを用いて右から左へと逆順に合成される流れとが共存していることがおわかりいただけるでしょうか。計算機科学の感覚では、これはまさに「上り」と「下り」の情報流が共存する状況を表しています。

**Int** 構成は、Girard が 1980 年代末に創始した双方向の情報流を伴なう計算の理論である**相互作用の幾何**（Geometry of Interaction）の核心部分に暗黙のうちにあらわれています。のちに、Abramsky らは、**Int** 構成を基礎とした相互作用の幾何の再構成と一般化を与えました。

勝股は、40 年以上前に Knuth が導入した**属性文法**と呼ばれるコンパイラ等で文脈自由言語の意味の記述に用いられる技法が、**Int** 構成を用いて明快に理解できることを示しました。属性文法ではプログラムの構文木の構造に沿って双方向の情報のやりとりがなされるのですが、ここでもトレース付きモノイダル圏と **Int** 構成が重要な役割を果たします（下図）。




特に、高階のデータ（プログラム）を属性に用いる属性文法は、トレース付きモノイダル閉圏（モノイダル閉圏はカルテジアン閉圏の直積をテンソル積に一般化したもの）の上の **Int** 構成により理解できます。トレース付きモノイダル閉圏 $\mathcal{C}$ では、$\mathbf{Int}\,\mathcal{C}$ への埋め込みが右随伴関手を持ち、このことから、高階属性を用いて双方向の情報流が一方向の情報流に帰着されることを説明できます。例えば、**Cpo** はそのような例を与えます。プログラムのコンパイル時の最適化技法であるプログラム変換においても、関連する結果が得られています。

## 5　おわりに

プログラム意味論においてなぜ圏論が重要なのか、という疑問に対する、現時点での私の答えは次のようになります：「プログラム意味論の本質は、プログラミング言語に内在する本質的な構造を捉え、その構造を保存する変換を与え分析することにある。その目的に適した数学の枠組みとしては、現状では圏論がもっとも適している。」我ながらあまり面白みのない答えですが、道具として圏論を使ってきた経験から、それほど外してはいないと思います。

全く意図していなかったのですが、**FB** のような単純な関数型プログラミング言語からはじめて、ラムダ計算と高階関数型プログラミング言語の意味論、そしてトレース付きモノイダル圏を用いたプログラム意味論と相互作用の幾何、というこの講義の流れは、実は、私自身の研究がこれまで進んできた道（の一番太い部分）に、ほぼ沿ったものとなっています。自分が重要だと思うこと、自然に話せると思うことを軸に話を組み立て書き進めていったら、いつの間にかこうなっていました。限られた時間と紙数ではありますが、この分野の面白さを多少なりとも感じ取っていただけたら幸いです。

参考文献には、日本語で気軽に手に取って頂けそうなものを中心に挙げています。私自身が書いたものでは、[3] がもっとも一般の方向けですが、この講義では飽き足らないという方には、[1, 2] をお試しいただければと思います（いずれも私のページからダウンロードできます）。これらの内容は、講義と一部重複しています。数理解析研究所の過去の数学入門公開講座のテキストでは、勝股先生 [5] と星野先生 [4] のものが、この講義の内容と深く関連しています。いずれも圏論は隠し味程度に留められていますが、この講義のあとで読んでみると発見があるかも知れません。[3] が収録されている本には、鈴木先生達のタングルの圏に関する解説 [7] や、勝股先生のモナドと計算効果に関する解説 [6] も含められています（鈴木・葉廣→長谷川→勝股の順に話がつながるように書かれています）。プログラム意味論について日本語で書かれた本には、[10] があります。圏論については、Mac Lane の古典的な教科書 [9] があります。最近の本では Leinster の入門書 [8] が良いと思います。

## 参考文献

[1] 長谷川真人, 再帰プログラムの意味論について, 数学 **59** (2007), 180–191.

[2] 長谷川真人, プログラム意味論とトポロジー - 再帰, 相互作用, 結び目, 日本数学会 2010 年度秋季総合分科会企画特別講演アブストラクト.

[3] 長谷川真人, プログラム意味論と圏論, 圏論の歩き方 第 4 章, 日本評論社, 2015.

[4] 星野直彦, 型無しラムダ計算とモデル, 数理解析研究所数学入門公開講座, 2013.

[5] 勝股審也, プログラミング言語の意味論. 数理解析研究所数学入門公開講座, 2007.

[6] 勝股審也, モナドと計算効果, 圏論の歩き方 第 5 章, 日本評論社, 2015.

[7] 鈴木咲衣, 葉廣和夫, タングルの圏, 圏論の歩き方 第 3 章, 日本評論社, 2015.

[8] T. Leinster, Basic Category Theory, Cambridge University Press, 2014.

[9] S. Mac Lane, Categories for the Working Mathematician, 2nd ed., Springer-Verlag, 1998 （邦訳： 三好博之, 高木 理 （訳）, 圏論の基礎, シュプリンガー・ジャパン, 2005 / 丸善出版, 2012）.

[10] 横内寛文, プログラム意味論, 共立出版, 1994.

# A 圏論の諸定義など

ここでは、この講義で必要となる圏論の諸定義や例についてまとめています。簡単のため、圏の大きさ（大きい圏・小さい圏、ユニバース等）については曖昧にして書いていますが、この講義の範囲では問題はありません。

## A.1 圏

**定義** 圏 $\mathcal{C}$ とは、

- 対象の集まり $Ob(\mathcal{C})$、
- 各 $A, B \in Ob(\mathcal{C})$ に対し、$A$ から $B$ への射の集まり $\mathcal{C}(A, B)$)、
- 各 $A \in Ob(\mathcal{C})$ に対し、射 $id_A \in \mathcal{C}(A, A)$、
- 各 $A, B, C \in Ob(\mathcal{C})$ に対し、写像

$$\begin{array}{rccc} \circ & : & \mathcal{C}(B,C) \times \mathcal{C}(A,B) & \to & \mathcal{C}(A,C) \\ & & (\ g,\ \ f\ ) & \mapsto & g \circ f \end{array}$$

の組で、結合律 $(h \circ g) \circ f = h \circ (g \circ f)$ （$f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, $h \in \mathcal{C}(C, D)$）および単位律 $id_B \circ f = f = f \circ id_X$ （$f \in \mathcal{C}(A, B)$）を満たすものです。$id_A$ を（対象 $A$ のうえの）**恒等射**、$g \circ f$ を射 $f$ と $g$ の**合成射**と呼びます。

**注意** 文献によっては、$\mathcal{C}(A, B)$ を $\mathrm{Hom}_{\mathcal{C}}\mathcal{C}(A, B)$ または単に $\mathrm{Hom}(A, B)$ と書くことがあります。また、$f \in \mathcal{C}(A, B)$ であることを、$f : A \to B$ や $A \xrightarrow{f} B$ などとも書きます。

**例**

- 対象が集合、射が集合の間の写像である圏 **Set**
- 対象が集合、射が集合の間の二項関係である圏 **Rel** （本文参照）
- 対象が体 $K$ 上のベクトル空間、射がベクトル空間の間の線形写像である圏 $\mathbf{Vect}_K$
- 対象が体 $K$ 上の有限次元ベクトル空間、射がベクトル空間の間の線形写像である圏 $\mathbf{Vect}_K^{\mathbf{fd}}$
- 対象が群、射が群の間の準同型写像である圏 **Grp**
- 対象が完備半順序集合、射が完備半順序集合の間の連続写像である圏 **Cpo** （本文参照）

諸定義

- 圏 $\mathcal{C}$ の射 $f : A \to B$ が**同型射**であるとは、ある $g : B \to A$ が存在して $g \circ f = id_A$ および $f \circ g = id_B$ が成り立つことをいいます。このとき、$g$ を $f$ の逆と呼び、$f^{-1}$ であらわします。また、対象 $A$ と $B$ の間に同型射が存在するとき、$A$ と $B$ は**同型**であるといい、$A \simeq B$ とあらわします。
- 圏 $\mathcal{C}$ に対し、**双対圏** $\mathcal{C}^{op}$ とは、$Ob(\mathcal{C}^{op})$ かつ $\mathcal{C}^{op}(A, B) = \mathcal{C}(B, A)$ であり、$\mathcal{C}$ と同じ恒等射、および $\mathcal{C}$ の合成射をひっくり返した合成射を持つものです。
- $\mathcal{C}$ と $\mathcal{D}$ を圏とするとき、$\mathcal{C}$ と $\mathcal{D}$ の**直積圏** $\mathcal{C} \times \mathcal{D}$ とは、$Ob(\mathcal{C} \times \mathcal{D}) = Ob\mathcal{C}) \times Ob(\mathcal{D})$ かつ $(\mathcal{C} \times \mathcal{D})((A, A'), (B, B')) = \mathcal{C}(A, B) \times \mathcal{D}(A', B')$ であり、恒等射 $id_{(A,A')} = (id_A, id_{A'})$ および合成射 $(g, g') \circ (f, f') = (g \circ f, g' \circ f')$ を持つものです。

## A.2 関手

**定義** $\mathcal{C}$ と $\mathcal{D}$ を圏とするとき、$\mathcal{C}$ から $\mathcal{D}$ への**関手** $F : \mathcal{C} \to \mathcal{D}$ とは

- 写像 $A \mapsto F(A)$ ： $Ob(\mathcal{C}) \to Ob(\mathcal{D})$
- 各 $A, B \in Ob(\mathcal{C})$ に対し、写像　$f \mapsto F(f)$ ： $\mathcal{C}(A, B) \to \mathcal{D}(F(A), F(B))$

の組で、$F(id_A) = id_{F(A)}$ および $F(g \circ f) = F(g) \circ F(g)$ をみたすものです。

諸定義

- 圏 $\mathcal{C}$ に対し、$\mathcal{C}$ 上の**恒等関手** $id_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ を、$id_{\mathcal{C}}(A) = A$、$id_{\mathcal{C}}(f) = f$ により定めます。また、関手 $F : \mathcal{C} \to \mathcal{D}$ および $G : \mathcal{D} \to \mathcal{E}$ に対し、**合成関手** $G \circ F : \mathcal{C} \to \mathcal{E}$ を、$(G \circ F)(A) = G(F(A))$ および $(G \circ F)(f) = G(F(f))$ により定めます。これらにより、対象を圏、射を圏の間の関手とする（大きな）圏 **Cat** が定まります。
- 関手 $F : \mathcal{C} \to \mathcal{D}$ は、
  - 各 $A, B \in Ob(\mathcal{C})$ について写像 $f \mapsto F(f)$ ： $\mathcal{C}(A, B) \to \mathcal{D}(F(A), F(B))$ が単射であるとき、**忠実**であるといいます。
  - 各 $A, B \in Ob(\mathcal{C})$ について写像 $f \mapsto F(f)$ ： $\mathcal{C}(A, B) \to \mathcal{D}(F(A), F(B))$ が全射であるとき、**充満**であるといいます。
- 圏 $\mathcal{C}$ が圏 $\mathcal{D}$ の**部分圏**であるとは、$Ob(\mathcal{C}) \subseteq Ob(\mathcal{D})$ かつ各 $A, B \in Ob(\mathcal{C})$ について $\mathcal{C}(A, B) \subseteq \mathcal{D}(A, B)$ であり、かつ $\mathcal{C}$ の恒等射および合成射は $\mathcal{D}$ のものの制限となっていることをいいます。このとき、$\mathcal{C}$ から $\mathcal{D}$ への包含関手を取ることができます。包含関手は常に忠実ですが、充満であるとは限りません。包含関手が充満忠実であるとき（すなわち各 $A, B \in Ob(\mathcal{C})$ について $\mathcal{C}(A, B) = \mathcal{D}(A, B)$ であるとき）$\mathcal{C}$ は $\mathcal{D}$ の**充満部分圏**であるといいます。

例

- 集合 $X$ に対し $X$ 自身を、写像 $f : X \to Y$ に対し二項関係 $J(f) = \{(x, y) \in X \times Y \mid f(x) = y\}$ を対応させる関手 $J : \mathbf{Set} \to \mathbf{Rel}$ は、忠実ですが充満ではありません。
- 集合 $X$ に対し完備半順序集合 $X_\perp$ を、$X_\perp = X \cup \{\perp_X\}$、$x \sqsubseteq y \Leftrightarrow (x = \perp_X$ または $x = y)$ と定め、写像 $f : X \to Y$ に対し連続写像 $f_\perp : X_\perp \to Y_\perp$ を $f(x) = x$ $(x \in X)$, $f(\perp_X) = \perp_Y$ と定めることで、関手 $(-)_\perp : \mathbf{Set} \to \mathbf{Cpo}$ が得られます。$F$ も、忠実ですが充満ではありません。
- $\mathbf{Vect}_K^{\mathbf{fd}}$ は $\mathbf{Vect}_K$ の充満部分圏です。

## A.3　自然変換

**定義**　関手 $F, G : \mathcal{C} \to \mathcal{D}$ に対し、$F$ から $G$ への**自然変換** $\alpha : F \to G$ とは、写像 $A \mapsto \alpha_A \ : \ Ob(\mathcal{C}) \to \mathcal{D}(F(A), G(A))$ であって、任意の射 $f \ : \ A \ \to \ B$ に対し $\alpha_B \circ F(f) \ G(f) \circ \alpha_A$ を満たすものです。$\alpha_A : F(A) \to G(A)$ を、$\alpha$ の $A$ 成分と呼びます。

諸定義

- 関手 $F : \mathcal{C} \to \mathcal{D}$ について、$F$ 上の**恒等変換** $id_F : F \to F$ を、$(id_F)_A = id_{F(A)}$ により定めます。また、関手 $F, G, H : \mathcal{C} \to \mathcal{D}$ と自然変換 $\alpha : F \to G$ および $\beta : G \to H$ について、**合成変換** $\beta \circ \alpha : F \to H$ を、$(\beta \circ \alpha)_A = \beta_A \circ \alpha_A$ により定めます。これらにより、$\mathcal{C}$ から $\mathcal{D}$ への関手を対象、関手の間の自然変換を射とする圏 $[\mathcal{C}, \mathcal{D}]$（文献によっては $\mathcal{D}^{\mathcal{C}}$）が定まります。この圏を、しばしば**関手の圏**と呼びます。

- 各成分が同型射となる自然変換を**自然同型射**と呼びます。自然同型射は、圏 $[\mathcal{C}, \mathcal{D}]$ の同型射に他なりません。関手 $F, G : \mathcal{C} \to \mathcal{D}$ について、$F$ と $G$ の間に自然同型射が存在するとき、$F$ と $G$ は**自然同型**であるといい、$F \simeq G$ と書きます。

- 圏 $\mathcal{C}, \mathcal{D}$ について、
  - $G \circ F = id_{\mathcal{C}}$ および $F \circ G = id_{\mathcal{D}}$ となる関手 $F : \mathcal{C} \to \mathcal{D}$ と $G : \mathcal{D} \to \mathcal{C}$ が存在するとき、$\mathcal{C}$ と $\mathcal{D}$ は**同型**であるといいます。
  - $G \circ F \simeq id_{\mathcal{C}}$ および $F \circ G \simeq id_{\mathcal{D}}$ となる関手 $F : \mathcal{C} \to \mathcal{D}$ と $G : \mathcal{D} \to \mathcal{C}$ が存在するとき、$\mathcal{C}$ と $\mathcal{D}$ は**同値**（**圏同値**）であるといいます。

## A.4　随伴関手

**定義**　圏 $\mathcal{C}$, $\mathcal{D}$ と関手 $F : \mathcal{C} \to \mathcal{D}$ および $U : \mathcal{D} \to \mathcal{C}$ について、関手

$$(C, D) \mapsto \mathcal{D}(F(C), D) : \mathcal{C}^{op} \times \mathcal{D} \to \mathbf{Set}$$

と、関手

$$(C, D) \mapsto \mathcal{C}(C, U(D)) : \mathcal{C}^{op} \times \mathcal{D} \to \mathbf{Set}$$

が自然同型であるとき、$F$ を $U$ の**左随伴関手**、$U$ を $F$ の**右随伴関手**であるといいます。

例

- $\mathcal{C}$ を有限直積を持つ圏とするとき、対象 $X_1, \ldots, X_n$ を直積 $X_1 \times \ldots \times X_n$ に写す関手は、対象 $X$ を $n$ 個の組 $(X, X, \ldots, X)$ に写す対角関手 $\Delta_n : \mathcal{C} \to \mathcal{C}^n$ の右随伴関手になります。

- $\mathcal{C}$ をカルテジアン閉圏とし、$X$ を $\mathcal{C}$ の対象とします。対象 $Y$ を $Y \times X$ に、射 $f : Y_1 \to Y_2$ を $f \times id_X : Y_1 \times X \to Y_2 \times X$ に写す関手 $(-) \times X : \mathcal{C} \to \mathcal{C}$ は、対象 $Z$ を $Z^X$ に、射 $g : Z_1 \to Z_2$ を $\overline{g \circ \varepsilon} : Z_1^X \to Z_2^X$ に写す関手 $(-)^X : \mathcal{C} \to \mathcal{C}$ の左随伴関手です。

- 関手 $J : \mathbf{Set} \to \mathbf{Rel}$ は、集合 $X$ を冪集合 $2^X$ に写す右随伴関手を持ちます。

- $\mathbf{Cpo}$ の対象 $X$ を $\mathbf{Int}(\mathbf{Cpo})$ の対象 $(X, 1)$ に、射 $f : X_1 \to X_2$ を $f \times id_1 : (X_1, 1) \to (X_2, 1)$ に写す関手 $J : \mathbf{Cpo} \to \mathbf{Int}(\mathbf{Cpo})$ は、$(X, U)$ を $X^U$ に写す右随伴関手を持ちます。