哲学 41 号（2023）

抽象化レベルの手法における 2 つの「抽象化」をデータベース的に理解する

榎本啄杜

1. はじめに

概念形成や抽象化という作用について考えるとき、いわゆる「アリストテレス的抽象主義」あるいは「古典的（伝統的）抽象主義」と呼ばれる立場が伝統的に受け入れられてきた[1]。これは、以下のように特徴づけられる立場である。

> 概念は複数のものに共通の性質を表す一般表象（徴標）であり、（それらの差異および類似性に注意を払い）差異を抽象（捨象）することにより形成される。したがって、抽象による概念形成は、常により特殊なもの（感覚・下位概念）を前提とするプロセスである。また、抽象によっては、すでに与えられていた特徴・性質が取り除かれるのみであり、新しい内容が付加されることはない。（浅野＆五十嵐 2021, p.121）

図 1 は「ポリピュリオスの樹」を簡易的に模したものだが、図中の「動物」に「理性的」という性質を付加すると、「動物」よりも下位概念である「人間」になる。一方で、図中において同じ抽象度である「動物」と「植物」の 2 つに共通する性質を抜き出すことによって、それらよりも上位概念である「生物」になる。このように、「上位概念に何らかの徴標（種差）が付け加えられることによって下位概念が得られ、逆に下位

[1] 浅野＆五十嵐（2021）によると、「アリストテレス的」という名称はカッシーラーに由来しているが、この立場の内実は、名称に反してアリストテレスに直接由来しているわけではない。

概念の共通部分を『抽象』することにより上位概念が得られる」（浅野＆五十嵐 2021, pp.121-122）。ここからもわかるように、伝統的な立場においては、「抽象化」という用語は「複数のものに共通する部分以外の性質は度外視したうえで、本質的[2]な性質を抜き出す」といったニュアンスで用いられてきた。

しかし、情報科学をはじめとする分野では、哲学における上述の理解とは異なる文脈で、このような伝統的な立場とは異なる意味で「抽象化」という用語が用いられていることが複数の論者によって指摘されている（cf. Colburn & Shute 2007; Ganascia 2012）。具体的には第 2 節以降で詳述するが、このような哲学の伝統とは切り離された、データベース的にも理解できるような意味における「抽象化」を自身の哲学理論に取り入れる哲学者が近年になって現れ始めた。そのうちの一人が、情報の哲学において多大な影響力を持つルチアーノ・フロリディである。

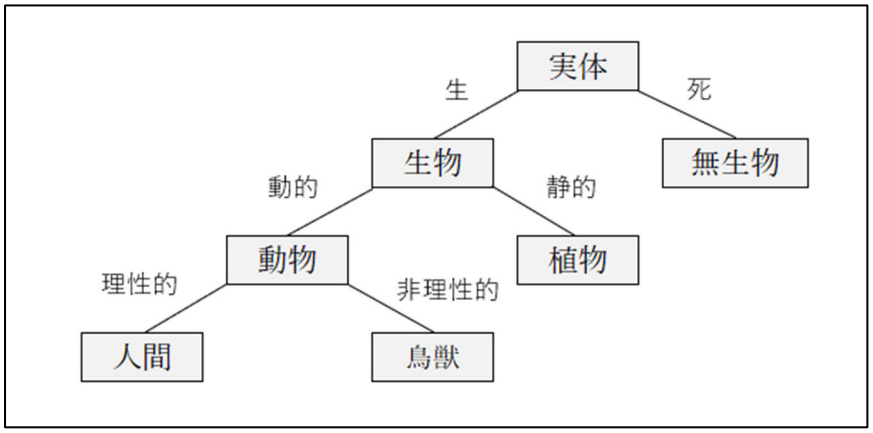

図 1　ポリピュリオスの樹

[2] 「抽象」と「本質」の関係については、モデリングについての伊理（2015）の論考が参考になる。以下の引用からは、「本質」を見出す行為がもっぱら認識論的な営みであることが読み取れる。「現実そのものではなく、そこに内在する本質的なものを取り出したものである。“取り出す（abstract）”とは“抽象”である。取り出すからにはそこに取り残されたものがあるわけであるから、“抽象”とはすなわち“捨象”である。[…] 『このことの本質はこれこれである』ということではなくて『私は、これこれがことの本質であると見るのだ』という、主体的な行動である。」（伊理 2015, p.36）

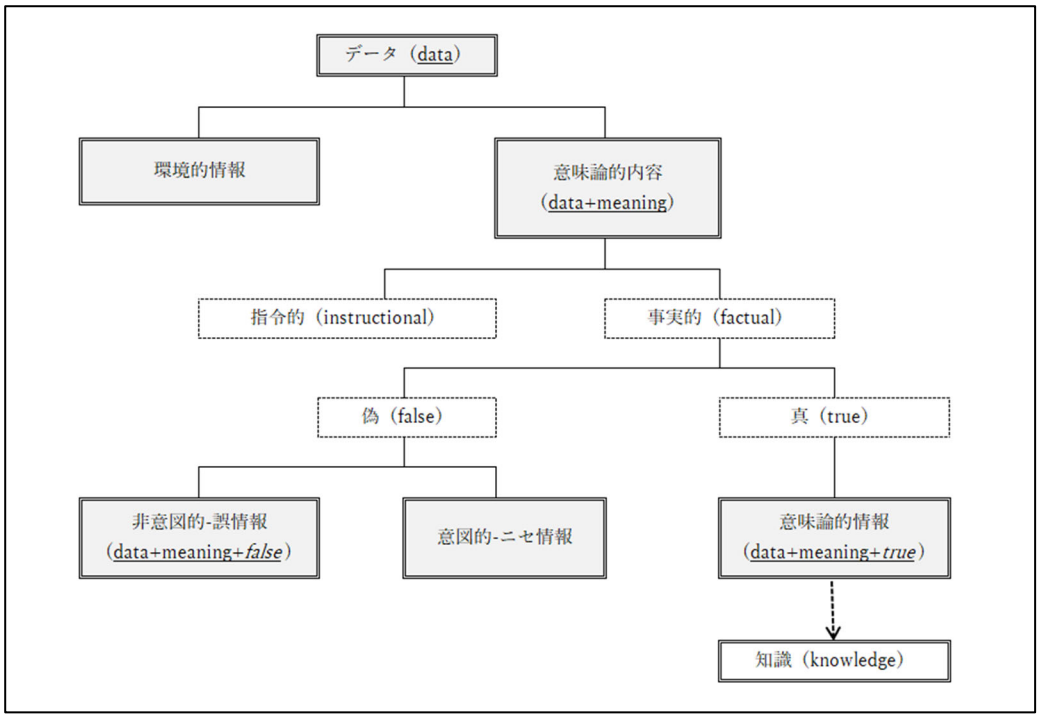

図 2　情報概念マップ（Floridi 2010 のものを筆者が加工）

フロリディは、自身が採用する主要な方法論として「最小主義（Minimalism）」「構築主義（Constructionism）」「抽象化レベル[3]の手法（Method of Levels of Abstraction）」の 3 つを掲げている（Floridi 2019）。その中でも特に「抽象化レベルの手法（以下、「LoA の手法」と呼ぶ）」は情報の哲学における主要な方法論として認められつつあり、当該分野の主要な問題群にも深く関わっていると言ってよい（Illari 2013）。たとえば、多義的な情報概念を見取り図形式で整理した「情報概念マップ」（図 2）や、フ

[3] 本稿では「Level of Abstraction」を一貫して「抽象化レベル」と訳している。筆者とは異なり「抽象の階層」と訳す先行研究（cf. 細川 2021; 山﨑 2022）もあるが、筆者は以下の理由により「抽象の階層」という訳し方を避けている。本稿第 3 節ではフロリディが 2 つの意味で抽象化概念を用いていることを指摘するが、階層的な入れ子構造を想定する「縦の抽象化」（3-3）とは異なり、「横の抽象化」（3-2）は階層的な入れ子構造ではない。日常的な「階層」という単語の用法を考慮すると、「抽象の階層」という訳語を選べば、フロリディの抽象化概念が「縦の抽象化」に限定されるかのような印象を読者に与えてしまうことが懸念される。「抽象化レベル」という訳が最善の訳だとも言えないが、なるべく先入観を与えないような訳語選択を心掛けた。

ロリディのマクロ情報倫理学（e-環境倫理）に登場する「情報圏（info-sphere）」「インフォーグ（inforg）」「デドメナ（dedomena）」といった主要概念は、LoA の手法を適用した結果得られるものである（Floridi 2011; 2013）。

しかし、その重要性に反して、LoA の手法自体を取り上げた先行研究は英米圏ですら少なく、十分に検討されているとは言い難い状況である。さらには、LoA の手法を解説しているフロリディ本人による文献（2-1 参照）には誤植や誤りが混在しており、未だ整備が必要な発展途上の段階だと言える。以上から、本稿で LoA の手法を検討することには、少なくとも次の 2 つの意義があると言える。第一に、情報の哲学における方法論を、今よりもさらに堅牢で明確なものへと鍛え上げられる。第二に、フロリディが用いる「抽象化」という用語は上述の伝統的な立場のものとは異なるため、この手法の有効性を示すことができれば、伝統的な抽象主義にとって代わる新たな「抽象化」概念を提示できる可能性がある。

本稿では、「（リレーショナル）データベース」との比較で LoA の手法を捉えることによって、その意義を首尾よく把握できることを指摘する。これを示すため、本稿は以下の構成をとる。まず第 2 節では、フロリディが LoA の手法を発展させてきた歴史的な経緯（2-1）を踏まえつつ、LoA の手法を筆者なりの仕方で再構成し、整理する（2-2）[4]。次に第 3 節では、LoA の手法における「抽象化」（3-1）が「横の抽象化」（3-2）と「縦の抽象化」（3-3）の 2 つに分けて捉えることができることを指摘し、この 2 つの抽象化がデータベースといかにして整合的に結びつくのか（3-4）を検討する。

2. 抽象化レベルの手法

本節の目的は、LoA の手法の開発者であり推進者でもあるフロリディ自身による説明を体系的に整理し直すことにある。フロリディは過去に LoA の手法に関する著作を

[4] 2-2 の内容は筆者による科学基礎論学会での発表（榎本 2022）が元になっている。

同名で複数執筆してきたが、その内容には若干のズレがある。執筆時期の違いによる内容の異同に由来する認識のズレを防ぐため、フロリディがどの時期にどのような点に重きを置いて執筆してきたかを 2-1 で確認しておく。それを踏まえたうえで、LoA の手法のエッセンスだと思われるものを、フロリディの意図に沿いつつ整理・再構成したものを 2-2 で提示する。

2-1. 抽象化レベルの手法の歴史的経緯

図 3 は、フロリディが執筆したもののうち、LoA の手法を主に扱っている文献のリストである。2004 年にジェフ・サンダースとの共著論文でその概要が示され、2008 年以降はフロリディの単著として出版されている。内容の大半は 2004 年と 2008 年で重なっているものの、タイトルの差からもわかるように、前者は「抽象化」という作用に着目しているのに対して、後者はその作用の結果として得られる「レベル」に関心がある点で異なる。この関心の差によって具体的にどのような違いが生まれているのかは、第 3 節（3-2 と 3-3）において詳述する。ひとまずこの段階では、2004 年と 2008 年でフロリディの関心の持ち方が異なるという点を押さえておけば十分である。

出版年	タイトル	備考
2004	The Method of Abstraction	共著論文
2008	The Method of Levels of Abstraction	単著論文
2011	The Method of Levels of Abstraction	*The Philosophy of Information* 収録
2013	The Method of Abstraction	*The Ethics of Information* 収録
2016	The Method of Abstraction	*The Routledge Handbook of Philosophy of Information* 収録
2019	The Method of Levels of Abstraction	*The Logic of Information* 収録

図 3　フロリディによる LoA の手法主要文献

では、2011 年以降に出版された残りの 4 本についても、フロリディの関心の持ち方をタイトルで判断してもいいのか（つまり、「Levels of」がタイトルに含まれているかどうかで分類してもよいのか）というと、事情が異なる。紙幅の都合により具体的な文章の差分を書き出すことはかなわないが、書籍に収録されている 2011 年以降の文献については、タイトルの差による強調点の違いは見受けられず、「レベル」に重きを置く 2008 年の方針を踏襲していると言える。言い換えると、タイトルだけで判断すれば 2013 年と 2016 年の 2 本は「抽象化」という作用に重きを置く 2004 年の方針を踏襲しているかのように見えるが、実際には 2004 年よりも 2008 年に近い方針で執筆されている。この点が、フロリディの議論をわかりにくいものにしている要因のうちの一つと言えるだろう。

以上の話をまとめる。LoA の手法に関するフロリディによる文献は 2019 年までに 6 本あるが、それらは

（i）「抽象化」という作用に重きを置く 2004 年の文献

（ii）その作用の結果として得られる「レベル」に重きを置く 2008 年以降の文献

の 2 種類に大別することができる。そしてこの違いは、第 3 節で述べるように、筆者が提案する「縦の抽象化」と「横の抽象化」というアイデアにそれぞれ大きく関わってくる。

2-2. 抽象化レベルの手法の概要

以上のように、LoA の手法関連文献は、執筆時期によって重きの置かれ方が若干異なる。しかし、その基本的な骨組みはどの時期においても共通しているため、以下で述べる LoA の手法の目的や定義は、時期を問わず共通したものだと考えてもらってよい。

まず、LoA の手法の目的を一言で言い表すならば、「物事を眺めるための土俵をどのように設定するか」についての方法を提供することである。一般に、生物は物事をありのまま写し取るわけではなく、抽象化という作用を介して認識する。図 4 による

と、システム（System）は分析される対象であり、その分析は特定の LoA においてなされる。さらに、この LoA によって生み出されるモデル（Model）こそが、フロリディの情報の哲学における主題である意味論的情報に等しい（Floridi 2019）。このモデル（つまり意味論的情報）はある特定の LoA に依存して生み出されているため、用いたレベルを無視して好き勝手に語ることはできない。あくまで、用いたレベルに関連する仕方において「このシステムは〇〇である」と語ることができる。

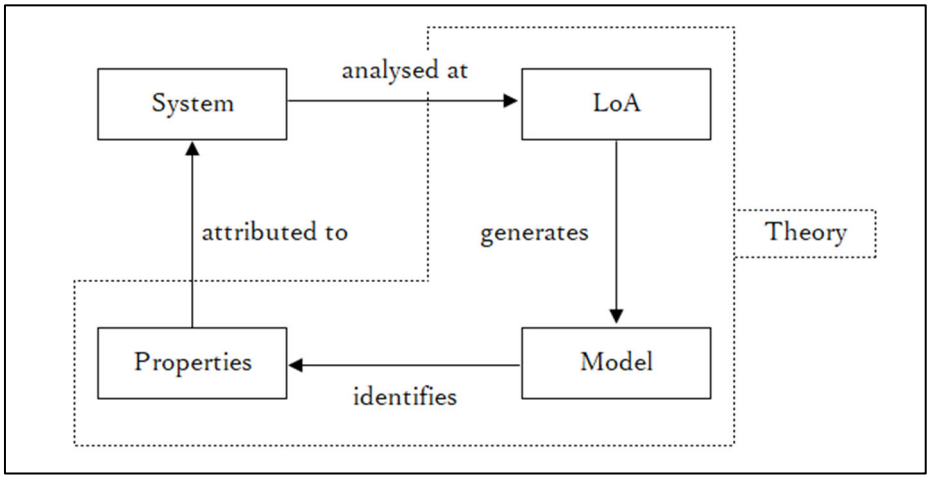

図 4　理論のスキーム（Floridi 2013）

2-2-1. オブザーバブルの集合としての抽象化レベル

では、対象を眺める土俵となる LoA はどのように定義されるのか。フロリディによれば、解釈された型変数であるオブザーバブル（observable）の集合が、LoA である（Floridi 2011, p.52）。キーワードとなる「型変数」、「解釈」、「オブザーバブルの集合」を順に解説する[5]。

[5] 解説の際には Microsoft Office のリレーショナルデータベースソフトである Access を用いる。LoA をデータベース的に解釈することが目的である本稿において、LoA のそもそもの解説をデータベースソフトで行うことは論点先取だと言えるかもしれないが、これはあくまで LoA の概要を一時的に把握するための便宜上の手段だと考えてほしい。

T00_社員

ID	社員番号	性別番号	都道府県番号	役職番号	部署番号	学歴番号	採用区分番号	考課番号
1	093	1	27	5	7	3	1	4
2	094	2	37	4	3	4	2	3
3	095	1	9	2	7	3	1	5
4	096	2	4	1	1	1	1	5
5	097	1	25	5	7	5	1	3
6	098	2	8	5	1	4	1	4
7	099	2	5	4	7	2	2	4
8	100	1	43	3	1	2	2	3
9	001	2	35	2	8	2	1	4
10	002	2	4	4	5	1	2	2
11	003	1	3	6	5	3	1	5
12	004	1	39	6	8	5	2	2
13	005	1	32	1	4	4	2	2
14	006	2	21	6	1	1	2	3
15	007	1	18	5	6	4	2	4

図 5　社員テーブル

T00_社員

フィールド名	データ型
ID	オートナンバー型
社員番号	短いテキスト
性別番号	数値型
都道府県番号	数値型
役職番号	数値型
部署番号	数値型
学歴番号	数値型
採用区分番号	数値型

図 6　社員テーブルのデザインビュー

例として、社員数 100 人の会社で、人事データを図 5 のような仕方で管理している状況を想定する。この社員テーブルで全社員を管理しており、そこには、社員番号が何番の人は役職番号が何番で…というように、各項目に何らかの値が入力されている。この社員テーブルのデザインビュー（図 6）を見ると、項目ごとにデータ型が定められていることがわかる。

したがって、各項目にデータを入力すれば、その項目にふさわしいデータ型が付与され、入力されたデータすべてが型付きの変数、つまり「型変数（typed variable）」としての資格を備えていることになる。たとえば、部署番号に入力されている数字が 1 であればそれは「数値型の 1」であり、社員番号に入力されている数字が 2 であればそれは「短いテキスト型の 2」になる。

順序組を用いると、型変数は二つ組

$$typed\ variable_i = <x_i\ ,\ type_i>$$

で表すことができ、2 つの型変数が等しいのは

$$typed\ variable_1 = typed\ variable_2$$
$$\Leftrightarrow\ <x_1\ ,\ type_1> = <x_2\ ,\ type_2>$$
$$\Leftrightarrow\ x_1 = x_2\ \text{かつ}\ type_1 = type_2$$

となる場合である。先の例で言えば、ある項目に入力された 1 と別の項目に入力された 1 は、変数としては等しい。しかし、前者のデータ型が「数値型」、後者のそれが「短いテキスト型」になれば、変数は一致しているもののデータ型が異なるため、2 つの型変数は等しくない。

さて、以上の型変数が「解釈（interpretation）」されると、オブザーバブルになるということだった。図 5 と図 6 によると、社員番号 096（ID は 4）の社員は、役職番号が数値型の 1 でありかつ部署番号も数値型の 1 である。しかし、型変数として等しいからといって、同じことを意味しているとは限らない。等しい型変数によって異なる対象を指示するためには、図 7 のように解釈用の別テーブルを用意し、型変数と解釈を紐づけなければならない。この解釈用テーブルによって、役職番号列に入力された数値型の 1 は「部長」を意味していると解釈され、部署番号列に入力されたそれは「人事部」を意味していると解釈されるようになる。

T04_役職

役職番号	役職名
1	部長
2	次長
3	課長
4	係長

T05_部署

部署番号	部署名
1	人事部
2	総務部
3	営業部
4	財務部

図 7　解釈用テーブル（役職番号テーブルと部署番号テーブル）

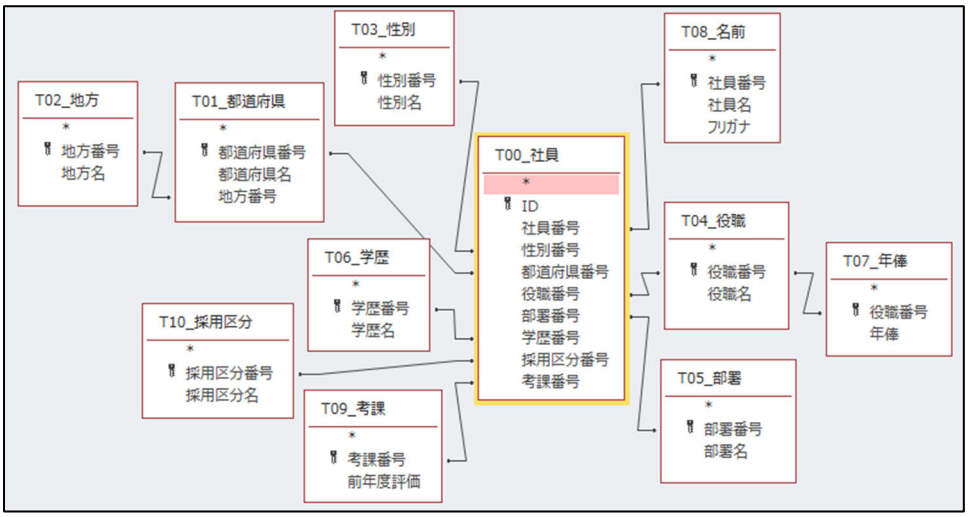

図 8　解釈の紐づけ（リレーションシップ設定）

これをすべての項目に対して行うと図 8 のように表すことができ、型変数が入力された図 5 の社員テーブルは、図 8 の解釈割り当てに基づいて指示対象を持つことになる。図 9 は、解釈がほどこされた後の社員テーブルである（当然ながら、各社員は架空の存在である）。図 5 でも見たように、図 9 の社員テーブルに入力されているデータは、元々は単なる数値であった。しかし、図 8 のようにそれぞれの項目に解釈を割り当てることによって、図 9 の段階では日本語表記のデータへと（一見すると）姿を変えた。これが「解釈された型変数」であり、LoA の構成要素であるオブザーバブルに他ならない。

Q01_全データ番号抜き

社員番号	社員名	フリガナ	性別名	部署名	役職名	年俸	学歴名	居住地
001	熊倉真奈	クマクラマナ	女	管財部	次長	7000000	修士・専門職	山口県
002	菊田李奈	キクタリナ	女	企画部	係長	5000000	博士	宮城県
003	児島清信	コジマキヨノ	男	企画部	課員	3000000	学士	岩手県
004	吉井利夫	ヨシイトシオ	男	管財部	課員	3000000	高卒	高知県
005	松島要一	マツシマヨウ	男	財務部	部長	8000000	短期大学士	島根県
006	三木三枝子	ミキミエコ	女	人事部	課員	3000000	博士	岐阜県
007	宇都宮恵一	ウツノミヤケ	男	広報部	主任	4000000	短期大学士	福井県
008	小澤正司	オザワマサシ	男	広報部	課員	3000000	博士	佐賀県
009	角田孝二	カクタコウジ	男	財務部	主任	4000000	短期大学士	埼玉県

図 9　解釈済みの社員テーブル

順序組を用いると、オブザーバブルは三つ組

$$observable_i = <x_i\,,\ type_i\,,\ interpretation_i>$$

と表すことができ、2 つのオブザーバブルが等しいのは

$$observable_1 = observable_2$$
$$\Leftrightarrow\ <x_1\,,\ type_1\,,\ interpretation_1> = <x_2\,,\ type_2\,,\ interpretation_2>$$
$$\Leftrightarrow\ x_1 = x_2 \text{ かつ } type_1 = type_2 \text{ かつ } interpretation_1 = interpretation_2$$

となる場合である。先の例で言えば、ある項目に入力された「数値型の 1」と別の項目に入力された「数値型の 1」は、変数とデータ型が共に一致するため、型変数としては等しい。しかし、解釈がそれぞれ「役職」と「部署」であれば、型変数としては一致しているものの解釈が異なるため、2 つのオブザーバブルは等しいとは言えない。

改めて言い直せば、LoA は「解釈された型変数」であるオブザーバブル（observable）の集合と定義されるのであった（Floridi 2011, p.52）。よって、ある特定の LoA は、以下のような集合

$$LoA_i = \{\, observable_1\,, observable_2\,, \ldots, observable_n\,\}$$

で表すことができる（ただし、n は 1 以上の整数）。たとえば、私があるシステムに対して「彼は〇〇部所属だ」と述べるとき、私は構成要素として「部署名」というオブザーバブルをもつ LoA に基づいて、そのシステムを眺めていることになる。なお、この集合は順序組ではなく、2 つの LoA が等しいのは、集合の構成要素が一致する場合である。

2-2-2．振る舞いによる調整

以上のように、LoA は構成要素にどのオブザーバブルを選択するかによって、異なるものとなりうる。しかし、今の段階ではどのようなものが構成要素になれるのかという制限がなく、任意のオブザーバブルを構成要素にとれることになってしまう。こ

れはつまり、対象を眺めるための土俵は何でもありで、好き勝手な認識が許されてしまうことに繋がりかねない。LoA の手法がシステムのモデリングに関わる以上、実際のシステムの振る舞いに応じて制限があってしかるべきである。

この何でもありな事態を防ぐために、オブザーバブルの一部である変数の振る舞い（behavior）を考慮して、上記の LoA を「調整された LoA（moderated LoA）」にしておく必要がある（Floridi 2011, p.53）。たとえば、現在例として用いている会社の社員数は 100 人であった。仮に、社員数は決して 100 人から増えることなく、社員の入れ替わりも決して起こらないとしよう。このとき、社員番号の振る舞い（とりうる範囲）は 001 から 100 であり、1 億をとることはありえない。また、他の値との組み合わせによる振る舞いの制限も考えられる。図 8 では「地方」と「都道府県」を紐づけているため、この 2 つの列における値は互いに影響を及ぼしている。あるシステムが大阪府に住んでいるのであれば、同じシステムが住んでいる地方は近畿地方でしかありえない（近畿地方を意味する値をとった場合、都道府県を一意に特定することはできないが、少なくとも近畿地方に存在する都道府県でしかありえないことはわかる）。

以降、単に「LoA」と言う場合には、構成要素のオブザーバブルがシステムの振る舞いを反映している「調整された LoA」のことを指すこととする。

2-2-3．抽象化レベルの集合としての抽象化勾配

LoA は物事を眺めるための土俵であるから、集合の要素となっているオブザーバブルの観点からのみシステムを捉えることができることになる。しかし一般に、我々は一つのシステムについて、一つの LoA からだけしか眺められないわけではない。あるシステムは部署名という観点からのみ眺められうるが、同時に、部署名は全く考慮に入れずに居住地という観点からのみ眺められることもあるだろう。このように、一つのシステムについて複数の異なる LoA で捉えることができるように、「抽象化勾配[6]（Gradient of Abstraction ; GoA）」という考え方が導入される（Floridi 2011, p.54-58）。

[6] LoA の集合がなぜ「勾配」と呼ばれているのかについては、p.49 で説明する。

抽象化勾配（以下、「GoA」と呼ぶ）は、LoA の集合として定義される。よって、ある特定の GoA は、以下のような集合

$$GoA_i = \{ LoA_1 , LoA_2 , \dots , LoA_n \}$$

で表すことができる（ただし、n は 1 以上の整数）。なお、$n = 1$ で構成要素が 1 つになるときは、その GoA は構成要素であるその LoA と等しい。LoA が GoA の特殊なケースであると定義できることからもわかるように、フロリディによれば、抽象化勾配の定義こそが抽象化の手法において根本的なものである（Floridi 2004, p.4）。

LoA の手法にとって GoA が重要であるのは、ある LoA から別の LoA へと移行する方法を、GoA が提供しているためである。ここで、新たに以下の語彙を使用する。ある LoA : L_n における変数の振る舞い（behavior）を p_n とすると、異なる 2 つの LoA : L_i , L_j 間において対応するオブザーバブルがあれば、そのオブザーバブルの各振る舞い p_i , p_j は関係 $R_{i,j}$ に基づいて、次のように翻訳できる。つまり、L_i における任意の振る舞い p_i は、L_i から L_j への関係 $R_{i,j}$ により、L_j 上の振る舞い $P_{R_{i,j}}(p_i)$ へと翻訳される（ただし、$0 < i \neq j$）。なお、R の逆は $R_{j,i}$ と表現することとする。このとき、GoA は次の整合性条件（consistency condition）を満たす（Floridi 2011, p.55）。

> 整合性条件：L_j 上の振る舞い p_j は、L_i 上の振る舞い p_i を翻訳した $P_{R_{i,j}}(p_i)$ と少なくとも同じくらい強い。つまり $P_{R_{i,j}}(p_i) \rightarrow p_j$ [7] である。

この関係性を図解すると図 10 のようになる。フロリディによれば、「各 LoA におけるオブザーバブルは、別の LoA におけるそれと明示的に関連づいていなければなら」ず、もしもこの整合性条件がなければ、「GoA を構成する各 LoA の振る舞い同士が繋

[7] 「強い」の意味するところは明確ではないが、おそらくは、翻訳先の述語の十分条件になることを最低限の条件として求めているものだと思われる。また、原書では論理結合子「→」の向きが逆になっている（$p_j \rightarrow P_{R_{i,j}}(p_i)$）が、誤植であると筆者は考える。というのも、$L_i$ 上の振る舞い p_i が関係 $R_{i,j}$ で翻訳されたとき、「→」の向きが原書通りだと、翻訳先によっては $P_{R_{i,j}}(p_i)$ でありながら振る舞い p_j を満たさない場合がありうる。これは筆者が作成した図 10 からも明らかであるように思われる。

がりを全く持たなくなってしまう」(Floridi 2011, p.55)。この点は 3-3 で「縦の抽象化」を考えるうえで重要になる。

最後に、GoA において最も重要な 2 つの特殊なケースを導入して、LoA の手法の説明を終わらせよう。つまり、「互いに素（disjoint)」な GoA と、「入れ子（nested)」の GoA の 2 つが GoA において重要である。

まず、互いに素な GoA について。一般に、GoA とは以下のように LoA の集合として定義されるのであった。

$$GoA_i = \{ LoA_1 , LoA_2 , \dots , LoA_n \}$$

ここで、構成要素である各 LoA の中で、共通したオブザーバブルを一切持たないとき、この GoA は互いに素であり、「関係 R はすべて空である」(Floridi 2011, p.56)。

一方で、L_i と L_{i+1} への関係 R が空ではなく、関係 $R_{i,i+1}$ の逆（つまり関係 $R_{i+1,i}$）が、L_{i+1} のオブザーバブルから L_i のそれへの全射となっているとき、この GoA は入れ子になっているという。

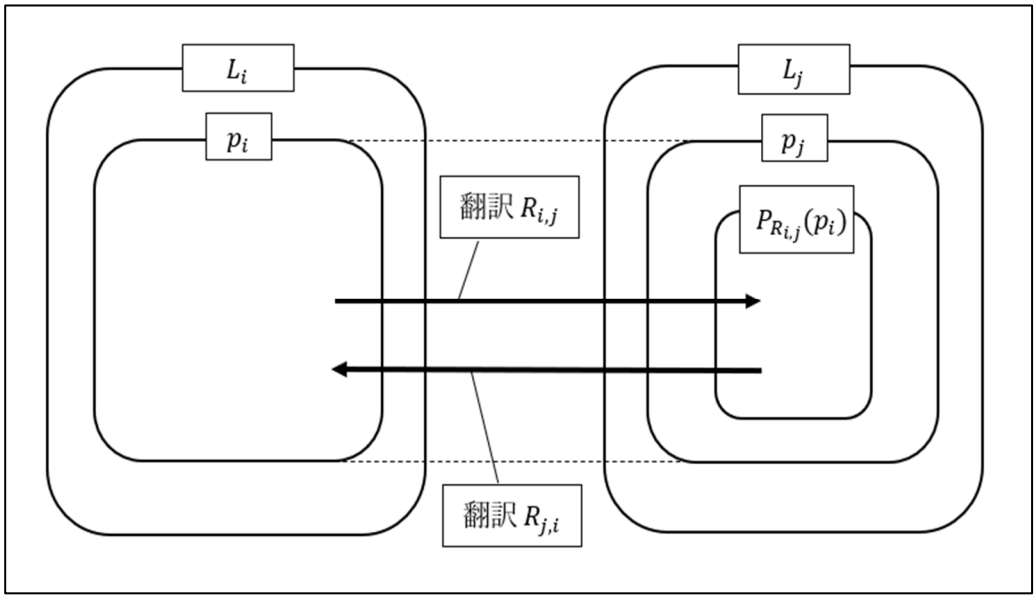

図 10　翻訳規則（関係 R）を集合で表した図

たとえば、社員の現住地だけを知りたい場合の LoA と、同じ社員の人事考課点が何点だったかだけを知りたい場合の LoA では共通するオブザーバブルがなさそうであるので、これら 2 つを構成要素にもつ GoA は互いに素だと言える。しかし、社員の現住地について、居住している都道府県について知りたい人と、居住している地方名について知りたい人は、粒度は異なるものの、ある程度対応したオブザーバブルを共有している。そのため、これら 2 つを構成要素にもつ GoA は入れ子だと言える。

3. データベースと 2 つの抽象化

前節では、LoA の手法についての再構成・整理を行った。それを受けて、本節では LoA を首尾よく理解するため、データベースの枠組みで捉えることを提案する。

まず、フロリディが「抽象化」という用語をどのように捉えているかを簡単に確認（3-1）した後、LoA は基本的にはデータベース的な発想で理解でき、筆者が「横の抽象化」と呼ぶものに相当することを指摘する（3-2）。一方で、筆者が「縦の抽象化」と呼ぶような意味での用法も無視できないことを確認（3-3）したうえで、それでもやはりデータベース的発想に還元して考えられることを主張する（3-4）。

3-1.「抽象化」の捉え方

LoA の手法においてキーワードとなるのは、言うまでもなく「抽象化」である。しかし、第 1 節でも述べたように、一般的に「抽象化」という用語は「具体化」と対をなすような仕方で、「複数のものに共通する部分以外の性質は度外視し、抽象度を上げていくことで本質的な性質を抜き出す」といったニュアンスで用いられることが多い。その点、「互いに素な GoA」が互いに共通する部分を持たない場合ですら「抽象化」と呼ばれていることからもわかるように、フロリディによる「抽象化」の使用法は、一般的な使用法から若干外れているように思われる。

フロリディによる定義を振り返ると、LoA とはオブザーバブルの集合であった。オブザーバブルが共通していようがいまいが、あるいは抽象度が上がろうが下がろうが、構成要素であるオブザーバブルの違いによって LoA が異なることになる。

図 11 はフロリディ自身が LoA の手法を説明する際に用いるものである。図のような状況であれば、2 つの LoA に共通するオブザーバブルを伴って次のように表現できるだろう。

$$GoA_1 = \{ LoA_1 , LoA_2 \}$$
$$LoA_1 = \{ observable_1 , observable_2 , observable_3 \}$$
$$LoA_2 = \{ observable_3 , observable_4 , observable_5 \}$$

これによると、ありうるオブザーバブルの集合（Space of observables）があり、その部分集合の選択が抽象化（Space of abstraction）だと考えられていることがわかる。つまり、どのオブザーバブルを選択・反映して、逆にどのオブザーバブルを切り捨てるか（サーチライトに喩えるならば、庭のうちどの部分を照らして、どの部分を照らさないか)、これを決めることがフロリディの想定する「抽象化」であると暫定的に言うことができる。

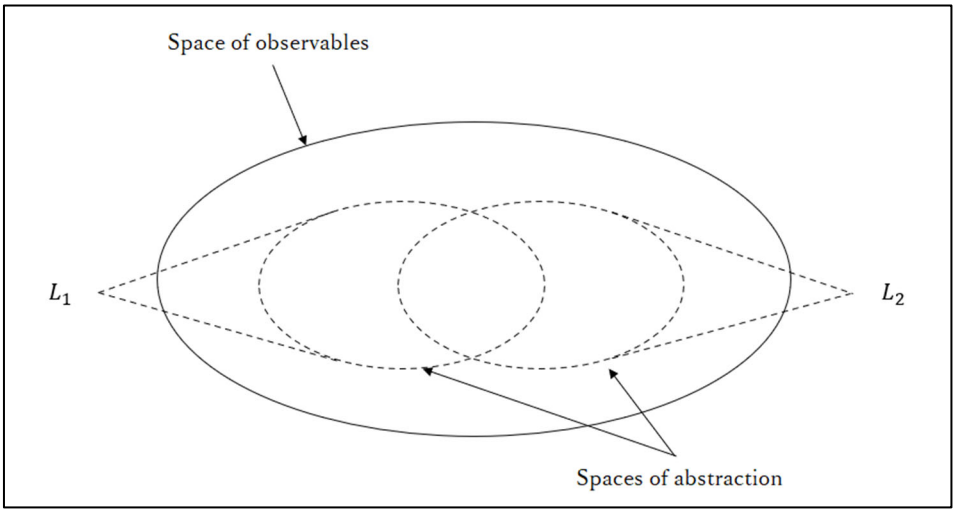

図 11　サーチライト図（Floridi 2011, p.77）

3-2．横の抽象化

フロリディ自身は LoA の手法について、モデル検査[8]に使用される形式手法（Formal Methods）から影響を受けていると述べている。ガナシアによれば、現代的な用語としての「抽象化」には複数の意味が目的や文脈に応じて与えられ、その中にはモデル検査的な文脈で使用されるタイプの「抽象化」も確かに存在する（Ganascia 2012）。それによれば、抽象化によってシステムを近似することで、問題を自動的に解けるレベルにまで単純化する。

しかし、フロリディの LoA はむしろデータベースにおける「ビュー（view）」概念に相当するもので、モデル検査的な「抽象化」とは異なるものだとガナシアは指摘している（Ganascia 2012, pp.8-9）。データベースはリレーショナル代数を用いて記述されるが、リレーショナル代数の諸演算を用いることによって、「実データベースの上にユーザーの視点に合った仮想的データベース空間を構築することができる」（増永 2017, p.187）。これがビュー機能である。フロリディ自身も LoA の例の一つとしてデータベースを挙げており（Floridi 2011, p.49）、ガナシアの指摘通りであれば、フロリディがモデル検査を例に出して LoA の手法を説明するというのは、ミスリーディングでさえあると言えるだろう。

実際、「データベースのデザインビュー画面においてどの項目を選択・反映するか」という操作は、ありうるオブザーバブルの集合のうちどのオブザーバブルを選択するかという操作と対応している。たとえば図 12 は、図 9 の解釈済み社員テーブルのデザインビューである。上から 4 行目に「表示」という機能があるが、この行にあるボックスのチェックを付け外しすることによって、結果として表示されるデータシートビューでの表示項目を変えることができる。つまり、ユーザーの興味や目的によってどの項目を選択・反映するかが自由に設定できるということであり、これが「仮想的」なデータベースと呼ばれる所以である。

---

[8] 「モデル検査」は、「形式仕様」が「モデル」を満足するかを自動的に検証する手法である。したがって、「モデル検査」はシステム開発の一部の工程を自動化できる。[…] 「モデル検査」では、「モデリング」「使用記述」「検証」の作業が必要になる。（赤間 2012, p.132）

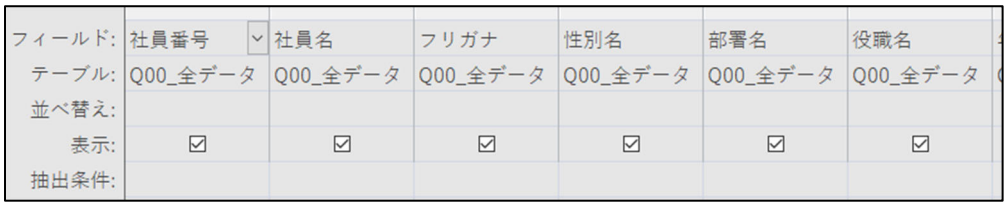

図 12　デザインビューのデザイングリッド部分

このように、LoA という発想における「抽象化」概念は、データベース的に捉えたとき、表示項目（オブザーバブル）を選択するという意味で捉えることができる。Colburn & Shute（2007）の表現を借りるならば、伝統的な立場における「抽象化」は、捨象される側の性質に関して、たとえば「本質的ではない」といったような否定的な価値判断が伴う「情報無視（information neglecting）」の操作として考えられていた。一方で、フロリディが用いるタイプの抽象化はそれとは異なり、本質とは独立して単に表示させる項目を選択するに過ぎない価値中立的な「情報非表示（information hiding）」の操作だと言うことができる。そこで、データベースにおける項目名は大抵横方向に並べられるため、本稿では便宜上、これを「横の抽象化」と名付けることとする[9]。すると、ガナシアの主張は「フロリディが用いる LoA という発想は、『横の抽象化』的な意味における抽象化に基づいている」と言い直すことができる。

なお、2-2-2 で問題となった各オブザーバブル間での組み合わせによる振る舞いの制限は、表示される項目間に存在する関数的な関係性として表現することができる。図 13 は、図 8 の一部を切り取ったものである。

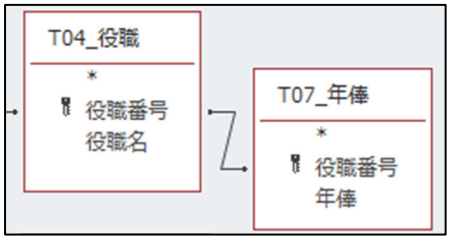

図 13　非入れ子型のタプル制約

[9] 特定の項目を選択して表示させる操作は射影演算であるが、射影の本質は横方向にあるわけではないため、「横の抽象化」という名称は満足なものとは言えない。

大半のテーブルは社員テーブルを解釈するために用いられているが、T07 の年俸テーブルは社員テーブルに紐づいておらず、T04 の役職テーブルにおける役職番号を解釈していることが読み取れる。これが意味するのは、役職番号の値が決まれば、年俸番号の値は一意に定まるということである（この場合は一対一対応になっており、逆に年俸番号の値が決まれば、役職番号の値も一意に定まる）。このように、互いに影響し合っているオブザーバブルの関係性をタプル制約として容易にデータベース上で表現できることは、LoA をデータベース的に捉えるというガナシアの主張をサポートするだろう。

3-3．縦の抽象化

フロリディの言う「抽象化」は筆者が呼ぶところの「横の抽象化」だ、というのがガナシアの主張だった。しかし、3-1 でも言及した「具体化」と対をなすような意味での「抽象化」を、フロリディは一切念頭に置いていないのだろうか。

図 14 の左半分は、図 2 の「情報概念マップ」の一部を切り取ったものである。

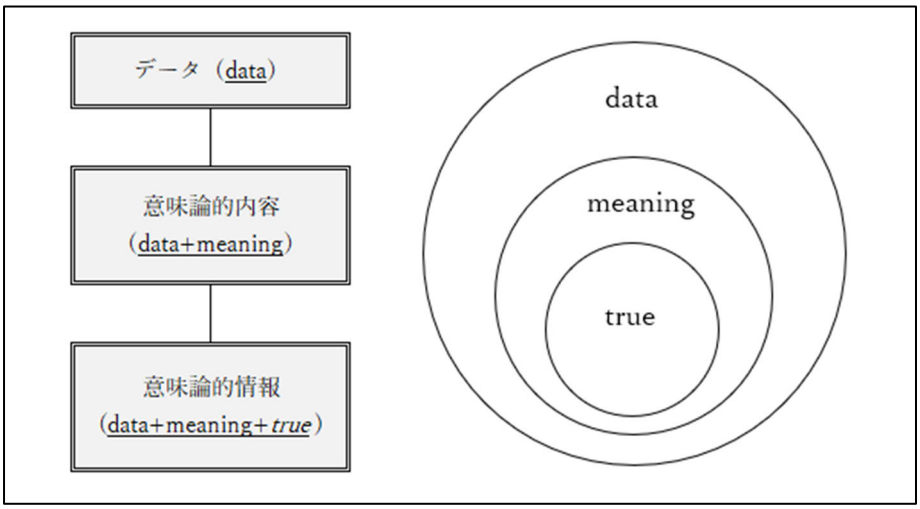

図 14　情報概念の階層構造

図 14 には「データ」「意味論的内容」「意味論的情報」と 3 つの情報概念が現れており、これらは階層構造をなしている[10]。フロリディはこれらの概念間の移行について「この移行は LoA の手法によって可能になる」と述べている（Floridi 2011, p.77）。このフロリディの発言からわかることは、少なくとも以下の 2 つである。つまり、

（1）アリストテレス的な伝統的抽象主義と同様、「抽象度を上げる」ことも抽象化であるが、それだけではなく、

（2）階層構造において性質を継承する場合、「親-子クラス間を行き来する」ことも抽象化

である。一般に抽象化とは（1）のように抽象度を上げる方向性でのみ語られるが、（2）によると、抽象度を下げる方向性も含めてどちらも「抽象化」だと考えられているように読み取れる。これは、本稿冒頭で言及した伝統的な抽象主義の「抽象によっては、すでに与えられていた特徴・性質が取り除かれるのみであり、新しい内容が付加されることはない」という特徴づけとは明確に異なる。むしろ、「新しい内容が付加されること」も積極的に受け入れられている。これをさきほどの「横の抽象化」と対比して、平面上では上下方向に行き来することを踏まえて、「縦の抽象化」と名付けることとする[11]。

ガナシアの指摘通り、確かに LoA は「横の抽象化」として捉えることができる。しかし、それでも LoA における「縦の抽象化」の意義を無視すべきではないと筆者は考える。図 3 では LoA の手法関連の文献をリストアップしたが、サンダースとの共著である 2004 年の”The Method of Abstraction”は、本稿 p.36 で説明した「整合性条件（consistency condition）」と「トップダウン構造（Top-Down construction）[12]」をキーワードとして、アブストラクトで次のように述べている。

---

[10] 図 14 の右半分は、図 11 のサーチライト図とは異なり、中央部に寄るにつれて解像度が高まっていき、より具体的なものになっていくものだと理解してほしい。p.49 の補遺でこの点を詳述している。

[11] 「横の抽象化」の場合と同様に、階層構造は縦方向に本質があるわけではないので、「縦の抽象化」という名称は満足なものとは言えない。

[12] 性質を継承する親-子の関係をもつ構造を意味する。

> オブザーバブルの集合［である LoA］の結果生まれる集まりは、「抽象化勾配」と呼ばれる。そして、それは選ばれた抽象化が必ず満たさなければならない最小限の「整合性条件」を形式化する（Floridi and Sanders 2004, p.1, 括弧内引用者）。

アブストラクトの段階で整合性条件について触れているのは 2004 年のこの論文だけであり、その後の文献では、トップダウン構造にもそれほどのこだわりを見せなくなった。

2-2-3 で各レベルの関係 R について最初に言及した際には、「各 LoA におけるオブザーバブルは、別の LoA におけるそれと明示的に関連づいていなければなら」ず、もしもこの整合性条件がなければ、「GoA を構成する各 LoA の振る舞い同士が全く繋がりを持たなくなってしまう」（Floridi 2011, p.55）と述べた。しかし一方で、GoA が互いに素であるときには「関係 R はすべて空である」（Floridi 2011, p.56）と述べられている。一見するとこの 2 つの言明は食い違っているように読み取れるが、整合性条件が元々階層（入れ子）構造を捉えるために導入されていたと考えれば辻褄が合う[13]。

つまり、整合性条件は親-子における継承関係が重視されていた 2004 年において、入れ子の GoA やそれに準ずる程度に共通するオブザーバブルがある GoA を捉えるために導入されており、筆者が呼ぶところの「縦の抽象化」をうまく記述してくれる。さらには、3-1 の冒頭で言及した、抽象解釈をはじめとしたモデル検査的な発想とも相性が良い。しかしながら、その後フロリディは筆者が呼ぶところの「横の抽象化」を重視する方針へ移行し、アブストラクトでの取り上げられ方からも推測できるように、整合性条件は以前ほどは重視されなくなってきた。2-1 でフロリディの関連文献を以下のようにグループ分けしたことを思い出してほしい。

（i）「抽象化」という作用に重きを置く 2004 年の文献
（ii）その作用の結果として得られる「レベル」に重きを置く 2008 年以降の文献

[13] むしろ、このように解釈しなければ「勾配（Gradient）」という表現と「整合性条件」を首尾よく理解することが困難になる。前者については p.49 の補遺を参照。

この 2 つは連続的ではあるものの、このうち（i）は「縦の抽象化」に重きを置いており、(ii）は「横の抽象化」に重きを置くようにフロリディの関心が変遷したと言い直すことができる。

3-4．2 つの抽象化をデータベース的に捉える

ここまでで、フロリディは「抽象化」という用語を 2 つの意味で用いている可能性があることを指摘してきた。言い換えると、基本的には「横の抽象化」として理解すればよいが、それだけではなく「縦の抽象化」の視点もある、という 2 つの側面があるものとして捉えた方が実態に即しているように思われる。

ここで、一つの疑問が生じる。データベース的に「抽象化」という用語を捉え直したものが「横の抽象化」であるため、当然、「横の抽象化」とデータベースは相性が良い。では、「縦の抽象化」はどうか？仮に「縦の抽象化」がデータベースと相性が悪いのであれば、LoA の手法を一律データベース的に理解することは、LoA の手法の都合の良い部分だけを取り出してデータベース的に分析しているということになりかねない。そのため、LoA の手法がデータベース的に捉えられると主張するのであれば、「横の抽象化」だけでなく「縦の抽象化」もデータベースの枠組みで捉えられることを示さなければならない。これは、「横の抽象化」だけに注目していたガナシアが見逃していた点である。

差し当たりの解決策としては、現実に存在する階層構造を反映するようにテーブル間のリレーションシップ設定を行うと、「縦の抽象化」を「横の抽象化」と同様のものとして扱えるようになる、というものが挙げられる。図 15 は、地方名と都道府県名が入れ子[14]になるように設定している様子である（一見すると図 13 と変わらないが、図 13 は非入れ子型として設定している点で図 15 とは異なる）。

[14] 「地方名」と「都道府県名」は入れ子の関係にあるのかという疑問が生じるが、緯度経度情報から考えれば両者は包含関係にあり、一種の階層構造をなしていると考えられる。

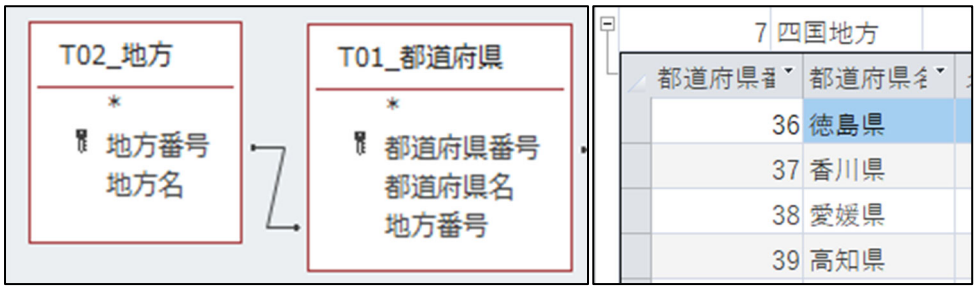

図 15　入れ子型のタプル制約とそのデータシートビュー

こうすることで、階層構造はデータベース上の射影演算として扱われることになり、「横の抽象化」が基礎的で、「縦の抽象化」はその特殊なケースとして解釈されるようになる[15]。つまり、階層構造という見た目の利便性を犠牲にすれば、「縦の抽象化」は「横の抽象化」に還元することができる。

以上より、フロリディが抱える 2 つの異なる抽象化、つまり「横の抽象化」と「縦の抽象化」は両者ともにデータベース的に捉えることが可能であることが示された。

4. おわりに

本稿の議論を簡単にまとめる。まず、フロリディやサンダースによって「抽象化の手法」「LoA の手法」と呼ばれる方法論が開発されたが、その内容や重きの置き方は時期によって若干異なるものであった（2-1）。本稿では、なるべく特定の時期の理解に偏り過ぎないように、LoA の手法の概要を筆者なりの仕方で再構成したうえで整理した（2-2）。次に、フロリディが用いている「抽象化（abstraction）」という用語が、古典的抽象主義や現代における日常的な理解とは異なるものだということを指摘し（3-1）、先行研究を踏まえながら、データベース的に理解すればうまく解釈できるというガナシアの主張を確認した（3-2）。筆者はこれを「横の抽象化」と名付けたが、一方で、フロリディの使用法には「横の抽象化」だけには限らない意味合いも含まれ

[15]「継承関係があるものも、実際には単に性質を付け外ししているに過ぎない」と考えれば、この結果は当然とも言える。

ているのであった。そこで、継承関係が存在するときには筆者が「縦の抽象化」と名付けたような解釈の余地を残しておくべきだと論じた（3-3）。最後に、「横の抽象化」と「縦の抽象化」の両方を同時にデータベース的に理解するための解決策、具体的には「縦の抽象化」を「横の抽象化」へと還元して考える案を提示し、先行研究の解釈がより広い範囲で適用可能であることを示した（3-4）。

筆者が考える本稿の意義は、以下にある。第一に、情報の哲学における方法論である LoA の手法を再構成・整理したうえで、そこには 2 つの異なる「抽象化」が同居していることを指摘し、両者に対してデータベース的な解釈を与えた（筆者はこれに「横の抽象化」「縦の抽象化」という独自のラベルを用いて整理した）。第二に、伝統的な抽象主義には本質を重視する（逆に、本質でないものを無視する）操作が見られたが、情報科学に影響された LoA の手法においては、価値中立的に表示項目を取捨選択する操作であることを指摘した[16]。抽象化の結果得られるものに「本質」のような認識論的価値が認められなくてもよい（し、認められてもよい）という点で、古典的抽象主義よりも射程が広く、同時に古典的抽象主義も説明できるような説明力を有すると言える。

ただし、以下の 3 点については今後の課題として残されている。第一に、注 5 でも述べたが、LoA の手法をデータベースで解釈したうえで解説する行為は論点先取であると同時に、読者の理解をデータベース的なものに過剰に制限してしまう恐れがある。データベースを用いないような、より一般的な説明手法が求められる。第二に、注 9 と注 11 でも述べたが、第 3 節で提案した独自のラベルである「横の抽象化」と「縦の抽象化」は、それぞれの本質が横方向や縦方向にあるわけではないため、必ずしも実態に即した満足なものとは言えない。あるいは、より深く理解することによって、縦／横の二分法で捉えること自体が誤りだと発覚する可能性もある。二次元的な理解

[16] 「一切の価値判断を伴わない」という意味ではない。何らかの価値は前提としているだろうが、抽象化の結果得られたものに本質が伴っていなくてもよい、という点で価値中立的だと述べられていることに注意されたい。なお、「本質のような価値を認めなければ抽象化とは呼べないのではないか」という批判がありうるだろうが、まさにそのような伝統的な暗黙の前提を切り崩す点にこそ、LoA の手法の新規性が認められる。言い換えれば、得られたものに本質的な価値があるかどうかの認定に LoA の手法は関与しない。

に囚われないような、より良い用語を考える必要がある。第三に、確かに入れ子の GoA を「縦の抽象化」として捉えることには視覚的な利便性があると言えるが、結局「縦の抽象化」を「横の抽象化」に還元して説明できるのであれば、ガナシアの整理に従って「横の抽象化」だけで説明する方が、説明力や単純性などの点で優れているとも考えられるのではないか、という懸念点がある。2 つの側面があるものとして考えなければならない必然性を、本稿では満足に指摘できていない。

補遺

2-2-3 で言及したように、LoA の集合を「抽象化勾配（Gradient of Abstraction ; GoA）」と呼ぶのであった。ここで、なぜ LoA の集合を表現するために「勾配（Gradient）」という表現が採用されたのかという疑問が生じる。一般に勾配とは「水平面に対する傾き」や「傾斜」といった意味で用いられるが、フロリディ自身は特にその理由を述べていない。以下、筆者による推測を述べる。

ガナシアは、勾配という表現について「誤解を招きかねない」と述べている（Ganascia 2012; p.5）。確かにガナシアの指摘通り、「横の抽象化」としての GoA や LoA を理解するうえでは、勾配という表現は適切ではないように思われる[17]。ではどのように理解すればよいだろうか。

GoA の初出は 2004 年の”The Method of Abstraction”である。3-3 でも述べたように、この論文においては親-子における継承関係が特に重視されており、階層構造が主な関心だった。そこで、例として図 16 のような階層構造を考えてみよう。同じ抽象度の高さの点の集まりを円で表現する。

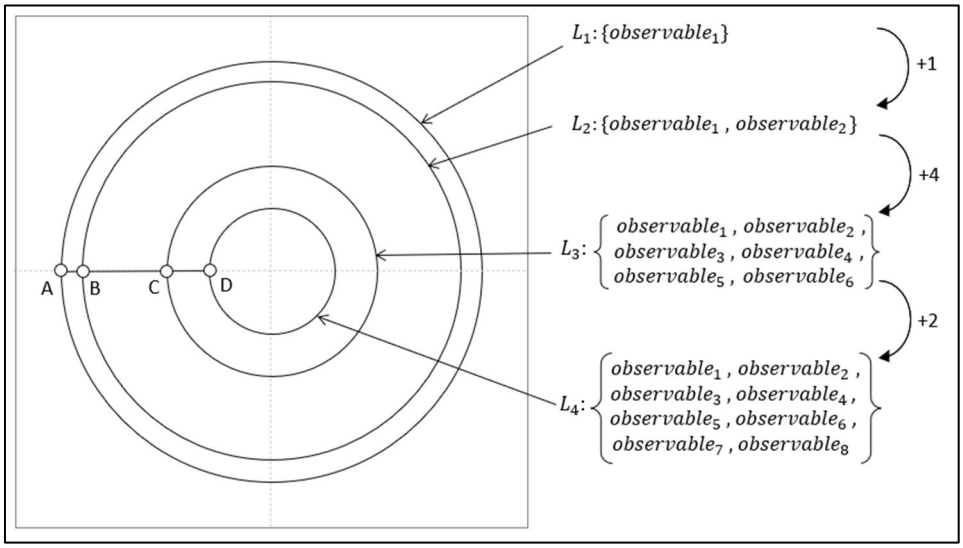

図 16　勾配の由来その 1

[17] ガナシアは勾配を物理学における「勾配ベクトル」だと理解したうえでこのように述べているが、厳密な専門用語としてではなく、日常的な用法として理解した際にも同様のことが言える。

このとき、オブザーバブルを 1 つしか持たないレベルをL_1とし、最も外側にある円周がL_1を示すものとする。そこから入れ子関係にあるオブザーバブルが増えるに従って、その増加分だけ一定の間隔で中心部へ向かって次のレベルを示す円が描かれていく状況を想定する。たとえば、L_2はL_1と比べてオブザーバブルが 1 つ増えているので、L_1よりも中心部に近く描かれている。この間隔を 1 とすると、L_3はL_2と比べてオブザーバブルが 4 つ増えているので、4 倍分だけ中心部に近く描かれる。また、L_4はL_3と比べてオブザーバブルが 2 つ増えているので、2 倍分だけ中心部に近く描かれる。言い換えると、線分$\mathrm{AB}:\mathrm{BC}:\mathrm{CD}=1:4:2$ である。なお、これはベン図的に表現される図 11 のサーチライト図とは異なる表現であることに注意されたい。図 16 では、外側から内側に進むにつれて構成要素であるオブザーバブル（ただし入れ子）が増えていき、それに伴い抽象度は低くなる。

ここで、各レベルを示す円は同じ抽象度の高さの点の集まりでできた線であることから、これを等高線とみなし、断面図に相当する図を作成する（図 17）。外側の円から順に抽象度が低くなっていくため、点 A’を通る直線が最も抽象度が高いL_1、点 B’を通る直線がL_2、点 C’を通る直線がL_3、点 D’を通る直線が最も抽象度が低いL_4である。GoA は各レベル間を移行するための手法であったから、各レベル間を線で繋ぐと、それぞれの抽象化度合いに応じた傾斜（つまり勾配）が生まれることがわかる。

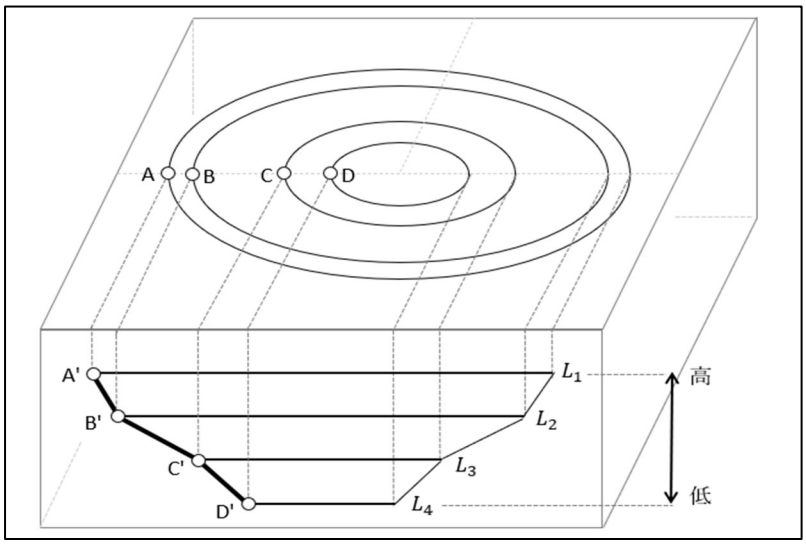

図 17　勾配の由来その 2

ただし、このように「勾配」という表現を理解するとき、以下のことに注意する必要がある。通常の等高線であれば、幅が広くなるほど傾斜は緩やかになる（幅が狭まるほど傾斜は急になる）。一方で、GoA を等高線的に表現した場合、幅が広くなるほど抽象度の高低差は大きくなる（幅が狭まるほど抽象度の高低差は小さくなる）。つまり、傾斜が緩やかであるほど抽象度の高低差が大きくなる（傾斜が急であるほど抽象度の高低差が小さい）ことになる。この結果は、勾配（傾き）の大きさと抽象度の高低差の大きさが逆関係になっており、一見すると反直観的である。これを整合的に解釈する方法は複数考えられるが、差し当たりは以下の 3 つが挙げられる。

（1）図 16 における線分 AB、BC、CD の長さの比を、抽象度の高低差の大きさとみなす方法

（2）図 18 のように、各レベル間を結ぶ線分と直交する直線の傾きの絶対値を、抽象度の高低差の大きさとみなす方法

（3）各レベルを示す直線と、各レベル間を結ぶ線分と直交する直線の 2 直線がなす角（ただし 90° 以下）の大きさを、抽象度の高低差の大きさとみなす方法

以上のような仕方で「勾配」を理解することができるのは、各レベルが階層構造をなし、GoA が入れ子となる場合だけである。抽象化の作用の結果として得られるレベルに重きを置く現在の LoA の手法から考えれば、確かに「勾配」は誤解を招きかねない表現である。しかし、主に「縦の抽象化」をうまく記述するための手法として考案された経緯を踏まえれば、「勾配」という表現への違和感は多少薄れるのではないか。

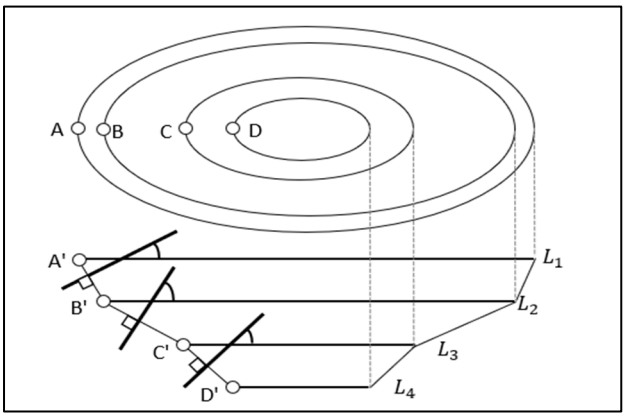

図 18　勾配の由来その 3

参考文献

Colburn, T. and Shute, G., 2007, “Abstraction in Computer Science”, *Minds and Machines*, 17, 169-184.

Floridi, L., 2008, “The Method of Levels of Abstraction*”*, *Minds and Machines*, 18, 303-329 .

Floridi, L., 2010, *Information : A Very Short Introduction*, Oxford University Press（塩崎亮・河島茂生訳, 2021,『情報の哲学のために　データから情報倫理まで』, 勁草書房）.

Floridi, L., 2011, *The Philosophy of Information*, Oxford University Press.

Floridi, L., 2013, *The Ethics of Information*, Oxford University Press.

Floridi, L., 2016, “The Method of Abstraction*”*, in L. Floridi (ed), *The Routledge Handbook of Philosophy of Information*, Routledge.

Floridi, L., 2019, *The Logic of Information*, Oxford University Press.

Floridi, L. and Sanders, J., 2004, “The Method of Abstraction*”*, in M. Negrotti (ed), *Yearbook of the Artificial, Nature, Culture and Technology*. Vol. 2: Models in Contemporary Sciences, Bern: Peter Lang, 177-220.

Ganascia, J, G., 2012, “Abstraction of levels of abstraction”, *Journal of Experimental & Theoretical Artificial Intelligence*, 1-14.

Illari, P., 2013, “What is the philosophy of information now?”, in The Π Research Network, *The Philosophy of Information – An Introduction*, 28-42.

赤間世紀, 2012,『形式手法　教科書』, 工学社

浅野将秀・五十嵐涼介, 2021,「カントとロッツェの抽象主義批判」, 池田真治編『抽象の理論をめぐる哲学史　古代から近代まで』,「抽象と概念形成の哲学史」研究会, 120-143.

伊理正夫, 2015,「モデリング」, 室田一雄ほか編『モデリング　広い視野を求めて』, 近代科学社, 35-44.

榎本啄杜, 2022, 「フロリディにおける 2 つの「抽象化」から考える概念の構造」, 科学基礎論学会ワークショップ「新しい「概念」の構造を求めて：概念論についての分野横断的研究」.

細川雄一郎, 2021, 「情報と変数」, 科学基礎論学会ワークショップ「変数から論理と数学を捉え直す：最近の成果」.

増永良文, 2017,『リレーショナルデータベース入門［第 3 版］　データモデル・SQL・管理システム・NoSQL』, サイエンス社.

山﨑紗紀子, 2022,「情報フロー理論と抽象の階層概念との統合に基づく情報の哲学の基礎的問題の解明」,『科学哲学』, Vol 55-1, 日本科学哲学会, 45-58