概念のインスタンス化と特異クラスについて

長谷部 陽一郎
同志社大学

1. はじめに

認知言語学では一般的に、言語形式によって媒介される概念のネットワーク的構造をタイプとインスタンスの概念を用いて説明する。しかしながら、多くの場合、異なった概念的粒度を持つタイプ間の関係と、タイプとその実体としてのインスタンスとの関係とを厳密に区別していない。つまり、異なった次元の問題を 1 つの次元に押し込んでしまっている観がある。

そこで本稿では、タイプ間の継承（inheritance）関係から、本来的な意味でのタイプ-インスタンス関係を独立させ、インスタンス化をタイプの実体化（reification）として捉え直すことを提案する[1]。さらにすべてのインスタンスについて、特異クラス（singleton class）—1 対 1 関係で結ばれる、各インスタンスに固有のクラス—を想定することが有効であることを論じる。これにより、制約ベースの形式主義的理論ではもちろん、認知言語学的な手法でも十分な説明が困難であったある種の言語現象に対する説明が可能になる。

本稿の提案は、計算機科学の領域（ソフトウェア工学）における、オブジェクト指向プログラミングの理論的発展に基づいている[2]。従来、認知言語学的な理論は、生成文法理論などに比べ、計算機科学との親和性が低いものと考えられてきた。しかし、それは正しくない。計算機プログラミングにはオブジェクト指向を始め数多くのパラダイムがあるが、それらの発展に重要な働きをしてきたのは、プログラムの記述に人間の自然な思考プロセスをどれだけ反映させられるかという問いであった。その意味で計算機プログラミングとはそもそも認知言語学的であり、その知見を採り入れることは、自然言語の理論をより精密化・洗練するにあたって有効であると同時に、計算機プログラミングという現実世界における経験的営みに裏打ちされた妥当性をもたらすことにつながる。

以下、次のように議論を展開していく。まず、2 節では、認知言語学におけるタイプ-インスタンス論の例として Langacker の認知文法理論におけるそれを概観し、問題点を指摘する。次に 3 節でこれに対する解決策として、継承と実体化のプロセスの明確な分離の提案および特異クラスの概念の導入を行う。また、特異クラスを実装したオブジェクト指向プログラミング言語である Ruby の設計の特徴についても簡単に触れる。4 節では 3 節での提案を受け、それらが実際の言語現象の説明にどのように役立つかをみる。扱うのは、固有名詞のインスタンス化および普通名詞のインスタンス化に関する問題である。最後の 5 節では全体のまとめを行う。

2. 概念のインスタンス化の捉え方に関する問題点

認知文法（Cognitive Grammar）[3]において、タイプとインスタンスの概念は理論の根幹を担うものとして重視されている[4]。しかし提示されている議論は抽象度が高いうえにかなり込み入っているため、全容を把握するのは必ずしも容易でない。ここではそれを敢えて簡略化して示し、問題のありかを際立たせてみたい。

しかしその前に、認知文法においてタイプとインスタンスの概念に関する議論が難解なものにならざるを得ない理由について触れておく。認知文法では、発話を実現するために、次の 4 つの意味関数（semantic function）が、表現の実質的内容に対して順に適用されなければならないとしている（Langacker 1991, Taylor 2002）。

(1) a. specification（具体化）
b. instantiation（インスタンス化）
c. quantification（数量化）
d. grounding（グラウンディング）

多くの場合、これらの意味関数を起動するための特別な語

彙グループが当該の言語内に存在する。英語の名詞句について言えば、形容詞は具体化の作用を担い、数量詞は数量化の作用を担う。また、冠詞や決定詞はグラウンディングのプロセスを起動する。例えば、the three big houses という名詞句の意味構造構築プロセスは、次のように表すことができる。

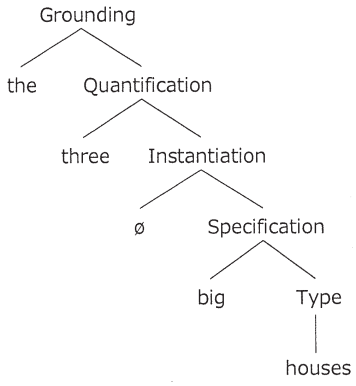

図 1

Taylor（2002, Ch. 4-5）によると、英語にはインスタンス化を担う特定の語彙グループが存在しない[5]。上の例で考えるならば、houses というタイプは big という形容詞により特定化された後にインスタンス化され、さらに three による数量化、the によるグラウンディングを受ける。しかしその際、インスタンス化のプロセスに関しては特定の語によって起動されるわけでない。それは純粋に「概念的な」プロセスとして存在する。形式的な反映を持たない概念的なプロセスに関する議論が抽象的になってしまうことは必然であろう。

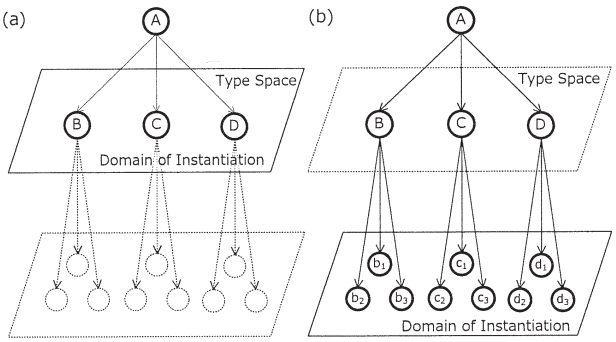

図 2

このような特徴を持ったインスタンス化というプロセスについて、Langacker（1991: 63）では、図 2 のような図式を用いて認知文法における考え方を示している。図 2a は、タイプとしての名詞の概念構造を示す。ここで C が（例えば）dog の概念を表しているとすれば、上位レベルの A はさしずめ mammal の概念である。dog とは、ある種の性質を備えた動物種の 1 つに対する名称であり、ある特定の個体だけに与えられた名称ではないため、C は階層の最下部ではなく、より上位のいわゆるタイプ・スペース（type space）に、他の動物種（哺乳類）B や D（例えば cat, mouse, etc.）と共に位置づけられる。

図 2a のような形で dog の概念が想起されるとき、それに属する個体の概念は、含意されはするもののインスタンスとして前景化されることはない（そのため階層の最下部は波線で描かれている）。しかしながら認知文法では名詞句の形成にはインスタンス化のプロセスが不可欠であることから、ここでは dog の概念自体が上位概念（mammal）のインスタンスとして扱われる。図中でタイプ・スペースとインスタンス化領域（domain of instantiation）が一致しているのはそのためである。

一方、図 2b は、Spot（あるいはコロでも良い）など、タイプ（ここでは dog）に属する個体に付けられた固有名の概念構造を示している。要素 A が mammal、要素 C が dog を表しているとすると、要素 c_1〜c_3 が Spot などの個体を表す。図 2b の場合、タイプ・スペースとインスタンス化領域とは別の階層に割り当てられており、前景化するインスタンス化階層に対してタイプ・スペース階層が背景化することになる。

言語形式を媒介として表現される意味構造は、何らかのレベルで実体化されなければならない[6]。そのためにインスタンス化というプロセスを想定したことは Langacker の卓見であると言える。しかし上の理論は、少なくとも、現象に対する説明力・記述力という点で明らかな限界がある。次にそのことについて述べる。

認知文法理論において、概念粒度の異なるタイプ間の関係とタイプ-インスタンスの関係とを同一軸上に展開しているのは適切でない。ここで概念粒度（conceptual granularity）とは、具体性ないしは非抽象性のレベルのことである。例えば mammal より dog の方が概念粒度

が高く、dog の個体である Spot はさらに高い。そこで図 2 で示されるような理論化がなされるわけであるが、タイプ-インスタンス関係は、本来、タイプ間関係と同じ概念粒度のスケール上に配置すべきものでないのである。

確かに認知文法ではインスタンスとして扱われる Spot という個体には、同種の個体には必ずしも見られない特性がいくつも備わっており、クラス要素 dog よりも粒度が高い。例えば、「ブチがある」「冒険心旺盛である」といった特性が認められることだろう。したがって、タイプ階層の上に固有名詞的概念としての Spot を位置づけること自体は正当である。しかし、そのような要素としての Spot はあくまで 1 つの「タイプ」であり、それ自体はインスタンスではない。インスタンスと呼ばれるべきは、タイプとしての Spot に対して 1 対 1 対応する要素（＝要素数 1 の集合の成員）として実体化された概念体の方である。タイプ間関係とタイプ-インスタンス関係を同一軸上へ展開することは、この事実を完全に見逃すことになる。

認知文法でも一応、個体のインスタンス化については特別な形式（c1, c2 のような小文字＋インデックス形式）で記述を行い、タイプ間関係との差別化を図っている。しかし、タイプ dog からインスタンス Spot が実体化されるという説明の方法はやはり問題である。同様にタイプ間関係についても、クラス dog はスーパークラス mammal のインスタンスであるとしているが、前者は後者を「実体化」したものではない。dog と mammal の関係は概念粒度の差に基づく継承関係であって、インスタンス化とは本来的に異なるレベルに属している。

さらに検討すべき別の問題もある。タイプ間の関係は必ずしも継承の関係に限られるものではなく、継承は多様なネットワーク的関係の一つの様式に過ぎない。例えば television という言語形式は、個体としての television（つまりテレビセット）タイプを喚起するほか、次のように多様なタイプを喚起する[7]。

(2) a. write for television
b. watch too much television
c. the invention of television

認知文法が採用している、タイプ間関係とタイプ-インスタンス間関係とを同一軸で捉えたモデルでは、その特性上、どうしても継承の関係に縛られてしまう。したがって、（2a）～（2b）のように、継承という観点からは結びつきが得られないが別のネットワーク的観点からはある種の結びつきが見出されるようなタイプ同士の関係を捉えることができない。このことからも、タイプ間の関係と、タイプ-インスタンスの関係とは別の系として扱うことが必要であると考えられる。インスタンス化のプロセスとは独立した系としてタイプ間の継承関係を含む様々なネットワーク構造を想定すれば、より柔軟な事例に対応することができるだろう。次の 3 節では、これを具体的にどのようにモデル化すべきかについて考えていく。

3. 継承とインスタンス化の分離[8]

前節で指摘した問題に対し本稿で提案するのは、1 節で述べたとおり、第 1 に継承関係と実体化との明確な区別であり、第 2 にすべてのインスタンスについて、それらと 1 対 1 で対応する特異クラスという概念体を想定することである。以下でその詳細について論じていくが、用いる用語の定義は認知文法を含む既存の認知言語学理論でのそれに必ずしも一致しないことに注意されたい。

一般的な前提として、自然言語の話者は、自らが心的に構築した概念を聞き手に再構築してもらうためのシンボルとして言語形式（すなわち語や句や文）を用いる。つまり、言語形式はある種の「型」すなわちクラスを想起するものであり、聞き手はこのクラスに基づいてインスタンスとしての概念を再構築するのである。当然ながら、話し手と聞き手とでは独立して概念構築が行われるため、自然言語によるコミュニケーションにおいては、話し手の意図と聞き手の理解と完全に一致するというのはあくまで理想であり、必ず達成されるとは限らない。しかし、ある言語コミュニティにおいては、言語形式と対応するクラスの知識が実質的に共有されているという前提のもとに、言語によるコミュニケーションは大きく破綻することなく成立する。

ここでクラスとは、共有が前提とされているという意味で静的（static）なものであり、インスタンスは発話の場においてリアルタイムで構築・再構築されるという意味で動的（dynamic）であるとすると、その関係は次のよう

に図式化できる。

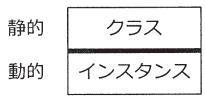

図 3

当然ながらクラスには様々なものがあり（少なくとも存在する言語形式の数よりは多いと考えてよい[9]）、それらは継承関係に代表される様々なネットワーク的関係を成す。このことを踏まえて、図 3 をより精細に描くと、次の図 4 のようになる。

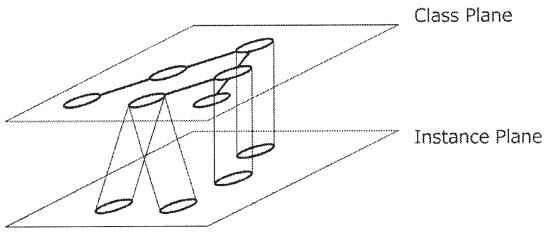

図 4

ここで前節において示した議論に立ち戻ると、図 2 のような認知文法におけるタイプ階層は、図 4 におけるクラス平面（class plane）におけるネットワーク関係の一形態であり、インスタンス化とはいわば直交する概念であることになる。認知文法のタイプ階層は基本的に概念間の継承関係に基づくものであり、そこでは「継承関係＝インスタンス化」というような含意があった。一方、図 4 では継承関係（およびその他のネットワーク的関係）をクラス平面内で完結させ、インスタンス化のプロセスをクラス平面からインスタンス平面（instance plane）への写像（mapping）として捉えている。クラス平面上の要素には様々な概念粒度のものがあり、必要に応じてそれらはインスタンスとして 1 対多の関係で実体化される。

詳しくは 4 節で見ていくが、このような構造を想定することにより、ある種の言語表現の意味構築・再構築の説明が容易になる。しかし、図 4 のような構造では説明が困難な表現も存在する。それは言語形式のシンボル的性質をメタ的な視点から捉えた言語表現、すなわち典型的には「X とは Y」といった表現の X 部を占めるような表現である。

先に発話がクラスとインスタンスの二層構造により成り立っているということを述べた（図 3 参照）。それは、言語形式と対応する静的なクラスが、発話の場の中で動的にインスタンスへと実体化されるという構造である。クラスが発話の参与者間で共有された知識構造だとすれば、「X とは Y」の X のようなクラス相当の概念体をリアルタイムで構築するためには、図 5 に示すようなクラスともインスタンスとも違う別の階層（クラス'）を想定しなければならない。

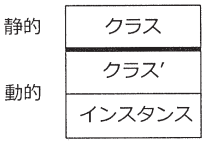

図 5

ここで強調したいのは、このようにクラス'という階層を新たに設けることは、上の X のような特定の言語現象を説明するための単なるアドホックな方略ではなく、自然言語に反映されるヒト認知の普遍的な性質に基づいているということである。自然言語の話者は、言語形式が担うシンボル性を、状況に埋没した主観的視点からのみならず、より「メタな」視点から認識することができる。ヒトは、言語形式 X によってある意味構造を構築・再構築できるだけではなく、そのこと自体を客観的に意識することができるのである[10]。そのような時に動的に想起される、言語形式と意味とのシンボル性に基づくメタ概念に着目し、それがインスタンス平面上に写像されるインスタンスに固有のものとして存在するクラスであることから、本稿ではこれを特異クラス（singleton class）と呼ぶ。

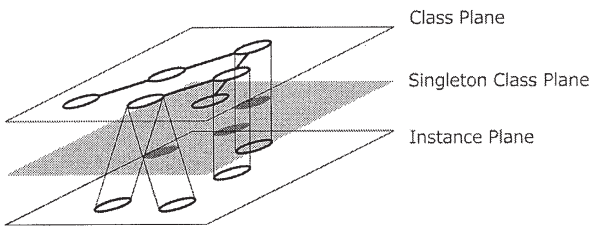

図 6

特異クラスの存在を考慮に入れて図 4 を修正すると、

図 6 のようになる。特異クラス平面はクラス平面とインスタンス平面の間に差し込まれる形で設定される。したがって、原則として全てのインスタンスはそれぞれに対応する特異クラスを持つことができる。ただしそれが実際にリアルタイムの談話の場において構築されるとは限らない。話者が言語表現のシンボリックな性質に対してメタな意識を働かせたときに初めて、特異クラス平面に特異クラスが構築される。ただし、一端それが構築された後は、静的な知識として通常のクラスへ組み込まれるときまでクラスの代理として働き続けるのである。

具体的な例を用いた説明は次節で行うということもあり、ここまでかなり抽象的な議論を続けてきた。重要な主張は、(1)概念間の継承関係と実体化のプロセスとは明確に分離すべきであること、(2)言語のシンボル的性質に対してメタな視点を向けた表現が可能という自然言語の特質を鑑みると、特異クラスという概念を導入すべきということ、の 2 点である。

以上はもちろん仮説に過ぎないが、少なくとも部分的には現実的な妥当性が保証された仮説である。なぜなら示されたメカニズムは、実際に広い用途に用いられている計算機プログラミング言語の設計と一致するものとなっているからだ。

プログラミング言語 Ruby では、オブジェクト指向（Object-Oriented）とよばれるパラダイムを可能な限り純粋な形で実現するために特異クラスの仕組みを導入している（まつもと・石塚 1999; 青木 2002; Thomas 2004）。Ruby ではリフレクション（refraction）[11]と呼ばれる機能が充実しているが、これも、徹底したオブジェクト指向的設計を追求した結果である。

より実際的には、Ruby の特異クラスは、特異メソッド（singleton method）と呼ばれる一種の「振舞」をオブジェクトに与えるために設けられた機構である。通常、オブジェクトの振舞は、それを生成する母型・鋳型であるクラスにおいてあらかじめ規定される。しかしながら Ruby では、すでにメモリ空間上に展開されたオブジェクト自体に動的に振舞を追加することができる。これは通常のクラス-インスタンス構造に基づいたモデルでは不可能なことである。メソッド（振舞）は、何らかのクラスに定義しなければならないが、通常のクラスにそれを定義してしまうと、そのクラスに属するすべてのオブジェクトが等しくメソッドを持つことになってしまう。そこで Ruby ではオブジェクトとクラスとの間に、そのオブジェクトだけに対応する新たなクラス（特異クラス）を必要に応じて差し込み、オブジェクト固有の振舞を実現する（図 7）[12]。特異クラスが構築されると、ポインタがつなぎ替えられて、オブジェクトは特異クラスのインスタンスとなる（青木 2002: Ch.4）[13]。

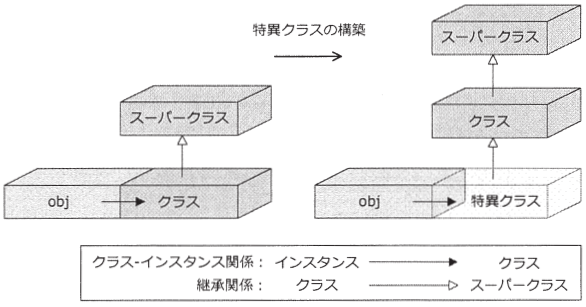

図 7

Ruby のオブジェクト機構のさらに興味深い点は、クラスのインスタンスだけでなく、クラス自体もまたオブジェクトとなっていることである。必然的に、クラスもまた 1 対 1 対応する特異クラスを持つことが可能である。下の図 8 は、クラス B、そのスーパークラスである A、そしてあらゆるクラスの継承元となっているクラス Object がいずれも各自の特異クラス（ここでは名前が括弧表記されている）のインスタンスとして構造化されている様子を示している。

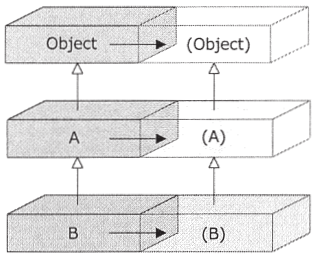

図 8

クラスに対する特異クラスの役割もまた、特異メソッドすなわち、そのクラス（＝クラス・オブジェクト）に固有のメソッドを実現することである。クラスもオブジェクトであるから情報（変数）を保持することができ、クラスが特

異メソッドを持てるということは、そのクラス固有の情報を取り出したり変更したりための振舞を定義できることを意味する。つまり、Ruby では巧妙な仕組により、本来は静的なものであるはずのクラスに対しても動的な操作を加えることができるようになっているのである。

以上のような Ruby（およびそれに類するプログラミング言語[14]）の実装上の仕組みは、先に論じた自然言語の構造についての仮説と呼応している。いずれの場合も、重要なのは、特異クラスを設けることで、任意のインスタンス・オブジェクトなりクラス・オブジェクトなりがカテゴリーの成員としてどのようなものであるかだけでなく、個としてどのようなものであるかをメタ的に述べることが可能になることである。次節では、自然言語においてそれが実際にどのように機能するのかという問題について論じていく。

4. 実際の言語現象への応用と考察

前節までの議論に基づいて本節で具体的に分析する対象となるのは、固有名詞のインスタンス化（4.1）と、普通名詞のインスタンス化（4.2）である。さらに本節では、概念構造における「継承」の位置づけという問題についても考察を加えていく（4.3）。

4.1 固有名詞のインスタンス化

固有名詞の意味構造については、認知文法でもかなり詳細な分析が示されており[15]、基本的な考え方には同意すべき部分も多い。しかし、2 節で示したような理論上の問題により、説明にアドホックな観があるのは否めない。Langacker は固有名詞の構造について次のように述べている。

> Because there is no distinction between type and instance at this level, a proper name is degenerate – its semantic pole is a type conception that is also an instance conception (i.e. it is a type with only one instance).
>
> (Langacker 1991: 62-63)

つまり、固有名詞の概念構造ではタイプとインスタンスの区別がもはや無く、固有名詞が意味として持つのはタイプであり、なおかつインスタンスである構造だということである。しかし、「タイプであり、かつインスタンスである」という言明が単なるレトリックでないとすれば、一体どのように理論的に示されるべきだろうか？

3 節で示した仮説に基づいて考えると、固有名詞（例えば犬の Spot）は、下のような構造として捉えられる。

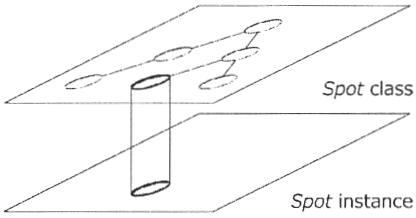

図 9

ここでいくつかの注意すべき事項を述べておきたい。Spot という固有名詞は基本的にその名前で呼ばれる対象のみに対応する。そこで、言語形式に対応する意味構造自体が「インスタンス」であると考えてしまいがちである。しかし、形式 Spot に直接対応するのはある種の「型」としてのクラス Spot であり、インスタンスとはそれがリアルタイムの発話の中で動的な記憶空間にマッピングされた（＝実体化された）結果である。

また、認知文法の枠組みでは図 9 のクラス平面に対応するクラス階層において、「固有名詞 Spot の直接上位概念は dog、さらにその上位概念は mammal」などと想定することが多いが、それはあくまで「あり得る」形の 1 つではあり、そのようなクラス階層が常に想定されるわけではない。すでに述べたとおり、クラス平面に展開するのは、継承関係を含む様々な種類のネットワーク関係であり、その瞬間の話者・聞き手の概念ネットワークにおいて最も関係する部分が局所的に展開されたものである。したがって、図 9 でのクラス Spot のスーパークラスは必ずしも生物種としての dog ではない。それは pet dog や picture-book character といった合成概念かもしれない。

このように、固有名詞の概念構造ではクラスとインスタンスが 1 対 1 対応すると考えられる[16]。ただしそれには「固有名詞の基本的かつナイーブな用法においては」という但し書きが付く。固有名詞のシンボル的構造をメタに認識した上でなされる用法については、新たな概念―特異クラスの概念―が必要になる[17]。

ここでいう固有名詞のメタ的用法とは、それによって、

特定の個体（およびその属性）を定義するような用法、すなわち、(3) ではなく、(4) のような用法である。

3. a. His mother finally found Spot at the end of the story.
   b. Spot wondered what was inside the box.
4. a. Spot is a puppy dog that appears in Eric Hill's picture book series.
   b. Spot lives with his mother.

つまりそれは、日本語で言えば「～は」「～とは」に接続する名詞句の用法である。そのような用法の主たる目的は、動的なクラス生成ないしは既存クラスに対する属性の追加・修正である。ただし、元来「共有されていると想定される知識」であるところの静的クラスを、リアルタイムで動的に構築ないしは修正することはできない。そこで、実際に構築されるのは、発話の場で実体化されたインスタンスに対する特異クラスであると考えるのである。（図 10 参照）[18]。

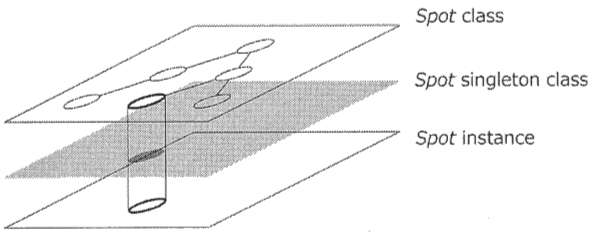

図 10

4.2 普通名詞のインスタンス化

次に、いわゆる普通名詞について考えてみたい。ただしここでは普通名詞を単に「固有名ではない名詞」というように広く捉える。少ない紙面で議論の要点のみを示すことになるため、名詞というカテゴリーの全てを網羅することは当然ながらできない。ここでは分析の対象を dog という可算普通名詞の概念構造に絞ることにする。

よく知られていることとして、英語の可算普通名詞は a + NP（a dog）の形で、(5a) のように純粋な個体を想起することもあれば、(5b) のようにカテゴリーを代表する、いわば観念的な個体を想起することもある[19]。

5. a. She has a dog with beautiful black hair.
   b. A dog makes a wonderful companion.

ここまでの議論に基づいて、(5a) の dog は図 11 のような構造のもとに発話されると考えられる。

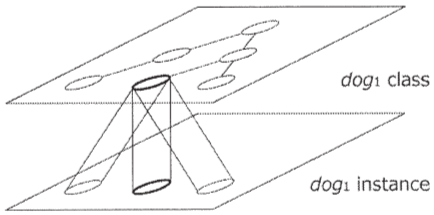

図 11

図 11 のクラス平面で強調を与えられている要素は、「個体としての犬」のクラスである。従来の理論では、「個としての犬タイプ」と「種としての犬インスタンス」を対立項として捉えることが一般的であった。しかし本稿で提示する枠組みでは、「個としての犬（dog_1）」と「種としての犬（dog_2）」はいずれもクラスであり、また実際の発話においてインスタンスとして実体化されるべき存在であると考える。

このような前提のもとにさらに（5b）のような用法の概念構造を考えると次のように図式化できる。

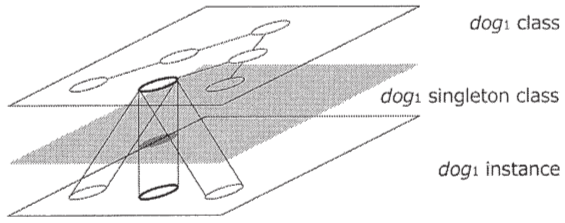

図 12

(5b) の dog の用法は「(1 匹の) 犬とは～」と解釈されるべきものである。それは単に個体としての犬のクラスのインスタンスを生成するのではなく、クラス自体の構成に働きかけ（再確認ないしは修正）を行う。したがって、「1 匹の犬」というクラスのインスタンスに対する特異クラスを想定すべきということになる。

それでは、普通名詞が「個としてのクラス（dog_1）」をインスタンス化するのではなく、「種としてのクラス（dog_2）」をインスタンス化する際にはどのような構造が

想定されるだろうか。これについては、名詞 dog(s)の次のような用法を用いて考えてみたい。

6　a. I like dogs more than cats.
　　b. The dog is a domesticated canine mammal.

(6a)の dogs のような可算普通名詞の複数形は意味論的にいわゆる物質名詞に近く、ここでの dogs は犬という種全体を一種の集合的概念として表す働きを担っている[20]。また（6b）でも the + NP の形式により、種としての dog クラスがインスタンス化されている。両者の違いは、(6a)が既存のクラスからの単純なインスタンス化であるのに対し、(6b)ではクラスを動的に再定義していることにある。これは（6b）ではインスタンスが通常のクラスではなく特異クラスによって実体化される構造を持つことを意味する。これを踏まえて（6a）および（6b）の図式化を行うと、それぞれ図 13 および図 14 のようになる[21]。

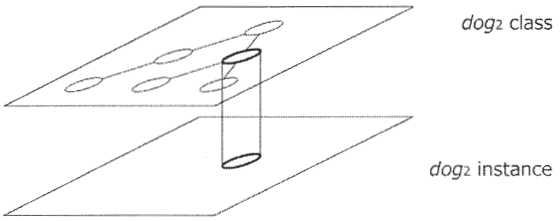

図 13

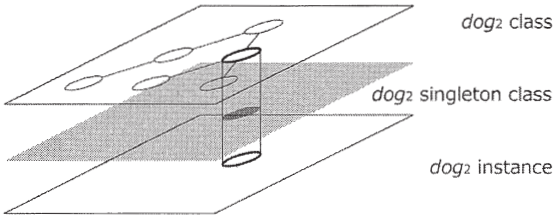

図 14

4.3 概念構造における継承の位置づけ

本稿では、タイプ間の継承をインスタンス化のプロセスから分離することと、特異クラスを想定することとを提案してきた。これにはさらに次のような利点がある。すなわち自然言語の概念構造を「例外に強く」「柔軟な」システムとして理論化できるということである。

そもそも継承とは何だろうか？自然科学的な観点から見たならばそれは、系統発生的な時間軸の上で種から種へと属性が引き継がれるプロセスである。あるいは、個体発生の連続において、より固有的な属性が親から子へと受け継がれるプロセスである。しかし、自然言語の概念構造のレベルにおける継承は、これら生物学的な継承と根本的に異なる。なぜならそれは構造自体が本質として持つものではなく、あくまで効率化のための高次の方策であるからだ。例えば animal – dog – golden retriever – Spot といった継承のラインは、あくまでひとりひとりの認知主体が「そのように」想定していることに過ぎない[22]。

したがって、概念構造の継承関係（およびその他のオントロジー的関係）は固定的なものとして扱うべきでない。それは一種の現実的な了解として有用である。しかしそのようなデフォルトの想定は常に覆され得るものであるため、変化を許容するシステムが求められるのである。

本稿で提案した理論ではクラス平面に展開されるネットワーク構造は継承関係のそれに限定されない。また、発話の各時点においては静的なものであるが、時間の経過とともに修正・変更され得る。ここで大きな役割を果たすのが特異クラスである。特異クラスは静的なクラス平面と動的なインスタンス平面を媒介し、クラス平面への新たなクラス定義を実現する。これらのことから、本稿のモデルは自然言語の概念構造が持つ「例外に強く」「柔軟な」性質を上手く反映できると考えられるのである。

5. まとめ

本稿の提案は、（1）継承関係とインスタンス化のプロセスの分離、（2）特異クラスの想定、であった。これらの提案の根拠となる事実と考察は上に示した通りである。ここでは、本論の中で前提として扱ってきたものの、必ずしも十分に論じることができていない事柄について触れてみたい。それは、「自然言語の理論の構築に、なぜ計算機科学やソフトウェア工学における知見が重要であるのか」という問題である[23]。

これについて最も重要な事実は、計算機プログラミング並びに計算機プログラミング言語の設計がヒトの認知システムのあり方に根ざした営みだということである。本稿で紹介した Ruby を含め、現在までに数多くのプログラミング言語が開発されてきた。チューリング完全であるそれらのプログラミング言語は、原理として同じアルゴリズム

を実装することが可能であり、どの言語で書かれたプログラムも同じ入力に対しては同じ結果を出力する。それにもかかわらず、なぜこれほどまでに多くのプログラミング言語が開発されてきたのか？

複雑な問題領域を明示的なアルゴリズムへと変換すること、さらにそれをコードの形式に移しかえるということは決して容易ではない。プログラミング言語は、いかにしてその困難なプロセスをヒトにとって自然で易しいものにできるか、効率的なものにできるかという問題意識のもとに改良されてきた。Ruby の設計に反映されているオブジェクト指向の考え方もそのような必要性から生じたパラダイムである[24]。

オブジェクト指向では、問題領域の様々な事象をそれぞれ 1 個の概念体として表現し、それらの相互作用を通じてプログラムとしての働きを実現する。オブジェクト指向のプログラミング言語への実装には様々なレベルがあるが、純粋なオブジェクト指向言語では、およそあらゆるものがオブジェクトとして表現される。これは認知文法ならびに認知言語学一般における考え方と親和性が極めて高いと言える[25]。

（認知）言語学の研究が、言語に反映されたヒト認知の仕組みを明示的な形式に抽出する作業であるとするならば、プログラミング言語の設計はそれを言語仕様に落とし込む作業である。両者とも、その過程にはヒトの認知に関する優れた理論が欠かせない。言語学が純粋な形でそれを追求できるのに対して、プログラミング言語の場合は必ずしもそうではない。しかし計算機上の実装系を持つそれには、ある意味で現実的な保証が与えられている。また常に進化競争にさらされた、淘汰と選択の対象となる存在であるため、絶えず漸進し続けていくことが期待できる。

以上のことから、認知言語学における理論のさらなる精緻化と洗練のためには、従来から言及されている生物学、脳科学、心理学などとの連携だけでなく、計算機科学およびソフトウェア工学との結びつきを強めていくことが重要であると思われる。本研究は、そのような研究プログラムの端緒となることを目指した、ささやかな実践である。

注

[1] ここで用いる reification という語の意味は、Langacker の認知文法でいうところの reification のそれとは異なる。

[2] オブジェクト指向プログラミングについては Mayer（2000）など数多くの参考となる資料・文献が存在する。

[3] Langacker (1997, 1990, 1991, 1999), Taylor (2002) などを参照。

[4] Langacker (1991)の Ch.2 は名詞の意味構造に関する章であるが、見方によってはタイプ-インスタンスの問題について丸々割かれた章と考えることができる。

[5] Taylor（2002）によれば、日本語や中国語のいわゆる類別詞（classifier）はインスタンス化に特化した語である可能性がある。この見方の妥当性に関してここでは議論しない。

[6] 詳しくは後述するが、本稿では、タイプ／クラスがあらかじめ知識として備わった概念体であるのに対し、インスタンスは発話時に動的に生起する概念体であると考える。

[7] 例は Taylor（2002: 462）から。

[8] 日本認知言語学会第 8 回大会での研究発表をもとに本稿を書き下ろす際、あらためて概念構造を記述・図式化する方法を検討したが、その際、次のサイトが大いに参考になった。

http://d.hatena.ne.jp/m-hiyama/20080109/1199863428

（檜山正幸「いまさらながらだけど、オブジェクトとクラスの関係を究めてみようよ」）

[9] ここで言語形式とは、もちろん何でも良いわけではなく、ある程度以上語彙化されたものである。

[10] これは Tomasello (1999)が論じた自然言語の起源論と呼応する見方である。生物学的な見地からすると、形式 ⇔ 意味というシンボリックな関係の把握は必ずしもヒト固有の能力ではない。ボノボやチンパンジーといった高等な霊長類にそれが可能であることは広く認められている。しかしシンボル関係に対してメタな視点を持つこととなると、ヒト以外の生物種にはきわめて困難なことである。これは、自らの心に対応するものとしての「他者の心」を自然に認識する能力に大いに関係する。

[11] プログラムの実行時にメモリ上に展開されたオブジェクトが自らの状態データを取得したり変更を加えたりできる機能をリフレクションと言う。これはあくまでメタフォリカルな表現だが、Ruby プログラムのオブジェクトは自分が何でありどのような状態にあるかを「知っている」。

[12] 図 6 および次の図 7 は青木（2002）の図式を一部改変して使用している。

[13] ただし Ruby では特異クラスの存在は、プログラマが「見ようとしなければ」見えないようになっている。したがってナイーブな視点からは、特異クラスが構築された後もオブジェクトはもともとのクラスのインスタンスであるかのように見える。

[14] 例えば、ここで見た Ruby のメカニズムは、1970 年代に開発が始まったオブジェクト指向言語 Smalltalk の影響を強く受けていると言われる。Ruby はそれまでのプログラミング言語の様々な機能や思想を貪欲に採り入れて作られたことで知られている。

[15] Langacker（1991）などを参照。

[16] 1 対 1 対応しない固有名詞の用法もある。ただしこの場合の

固有名詞は限りなく普通名詞に近い（下の例はいずれもインターネット上の blog 記事より）。

i. a. He was not the Bill Cowher that led the team to the Super Bowl in 2005-2006.

b. I am not a Bill Gates (who can solve a whole continent's problem alone).

[17] 注 13 で触れたように、Ruby では通常のプログラマの視点から見える構造と、さらに実装よりの視点から見える構造との 2 層構造が実現されているが、これは自然言語の構造と近いと考えられる。自然言語の話者は形式と意味との単純なシンボル的構造を主観的に使用するだけでなく、必要に応じてそれらに対し客観的・分析的な視点を持つことができる。この事実を明確な仕組みとして理論化することが、特異クラスを想定することの 1 つの重要な動機である。

[18] 概念の継承関係やタイプ-インスタンス関係だけでなく、談話の時間軸上における概念構造発達のモデル化という観点からも、このような仕組みは有用であると考えられる。

[19] ここでは a dog や、後で扱う the dog といった名詞句全体の（グラウンディング済みの）概念構造を扱うのではないことに注意されたい。分析の対象はあくまで a dog や the dog といった名詞句の一部としての要素 dog（ないしは dogs）におけるインスタンス化の構造である。

[20] dogs のような接辞付きの語を 1 つのクラスに対応させることについて疑問を感じる向きがあるかもしれない。しかしクラスとは決して認知文法で言うところの音韻極（phonological pole）に対応する概念ではなく、むしろ意味極（semantic pole）に対応する概念である。したがって、音韻的な分節性とは完全に独立している。Bolinger は「音形が異なれば意味が異なる」と論じたがその逆は必ずしも真ではない。

[21] 単数形 dog と複数形 dogs では、後者が概念内部の集合性を喚起するという意味で前者とは実際には異なるクラスを想起すると思われるが、ここでは説明の簡略化のため dog_2 として共通化する。

[22] もちろんそのような想定が文法や語彙に反映され固定化するということはあり得るが、その場合でも概念ネットワーク上での位置付けは常に変化し得る。「トマトは野菜であるか果物であるか」といった議論はその良い例である。

[23] あまり一般的に知られていないように見受けられるが、Goldberg（1995）には、構文文法における継承の概念がオブジェクト指向プログラミングにおける継承の概念から着想を得たものであると記されている。なお、オブジェクト指向プログラミングと認知言語学の親和性の高さについて論じた文献として Hasebe（2004）がある。

[24] オブジェクト指向以外のプログラミングのパラダイムについては Sethi (1996) や Van Roy and Haridi (2004) などを参照。

[25] 例えば認知文法では、概念構造はすべて「モノ」（thing）と「関係」（relation）に 2 分可能である。そしてモノと関係は「要素」（element）の下位範疇である。「要素」と「オブジェクト」とが対応すると考えれば両者の相似性は明らかである。

参考文献

青木峰郎. 2002. 『Ruby ソースコード完全解説』 東京: インプレス.

Goldberg, Adele E. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago: University of Chicago Press.

Hasebe, Yoichiro. 2004. Computer analogy reconsidered from a perspective of cognitive linguistics and object-oriented programming. *Proceedings of the Fifth Annual Meeting of the Japanese Cognitive Linguistics Association,* 126-36. [http://yohasebe.com/files/documents/JCLA2005Paper.pdf]

Langacker, Ronald W. 1987. *Foundations of Cognitive Grammar,* Vol. 1*, Theoretical Prerequisites*. Stanford, CA: Stanford University Press.

Langacker, Ronald W. 1990. *Concept, Image, and Symbol: The Cognitive Basis of Grammar.* Berlin: Mouton de Gruyter.

Langacker, Ronald W. 1991. *Foundations of Cognitive Grammar,* Vol.2*, Descriptive Application*. Stanford, CA: Stanford University Press.

Langacker, Ronald W. 1999. *Grammar and Conceptualization*. Berlin: Mouton de Gruyter.

まつもとゆきひろ・石塚圭樹. 『オブジェクト指向スクリプト言語 Ruby』 東京: アスキー.

Meyer, Bertrand. 2000. *Object-Oriented Software Construction*, 2nd Edition. Englewood Cliffs, NJ: Prentice Hall.

Sethi, Ravi. 1996. *Programming Languages: Concepts and Constructs*, 2nd Edition. Reading, MA: Addison Wesley.

Taylor, John R. 2002. *Cognitive Grammar.* Oxford: Oxford University Press.

Thomas, Dave. 2004. *Programming Ruby: The Pragmatic Programmer's Guide*. Raleigh, NC: Pragmatic Bookshelf.

Tomasello, Michael. 1999. *The Cultural Origins of Human Cognition*. Cambridge, MA: Harvard University Press.

Van Roy, Peter and Seif Haridi. 2004. *Concepts, Techniques, and Models of Computer Programming*. Cambridge, MA: MIT Press.

<abstract>

Redefining the Instantiation of Elements in Conceptual Networks

Yoichiro Hasebe
Doshisha University
yhasebe@mail.doshisha.ac.jp

The concept of "instantiation" in Cognitive Linguistics/Grammar is radically redefined by proposing a rigid separation of the deployment of hierarchical (and other kinds of ontological) structures of conceptual types on the Type Plane, and their instantiation process onto the Instance Plane. In the model proposed by Langacker (1991), relationships among types of different granularities and those of reification between types and instances are both rendered onto a single hierarchical tree structure. This is not considered appropriate, however, when considering that a theory should allow a flexible but relatively static nature of network configuration on the Type Plane while describing a more dynamic nature of the instantiation process.

In order to make possible an alternative theory of instantiation which is both cognitively and computationally realistic an analogy from the field of software engineering is adopted. Implementation techniques of an object-oriented programming language called Ruby are introduced. The natural outcome of a close comparison between a cognitive linguistic model of natural language and that of highly advanced computer language design is the proposal of a class-based instantiation model, which can replace the previous type-based one. Also proposed as a result of inspiration from the architecture of Ruby is the idea that an additional layer of the class-instance structure is assumed to take place right between the two planes. The additional layer, the Singleton Plane, is the place where dynamically created classes are deployed and respectively associated to one, and only one instance on the Instance Plane. By assuming singleton classes in addition to the original classes and their instances, construction of a model of natural language capable of explaining not only a subjective or "naive" way of using linguistic symbols but also a more objective or "meta" usage becomes more realistic. This new hypothesis is tested on linguistic phenomena of various instantiations of an English nominal "dog" and has been shown to be effective for actual linguistic research.

Discussed at the end of the paper is a possible question relating to why closing the gap between software engineering and cognitive linguistics is considered important. It is strongly argued that since computer programming is an art based on the basic cognitive abilities of human beings, concepts proved highly useful in that field—especially in object-oriented programming studies whose basic philosophy is highly topological to that of cognitive linguistics—are very likely to take a natural position alongside the study of natural language.