依存型意味論とオントロジーを用いた論理的多義とコアーション現象の分析にむけて

木下 恵梨子 [1,a]　中村 絢子 [2,b]　峯島 宏次 [3,4,c]　戸次 大介 [1,4,5,d]

[1] お茶の水女子大学 理学部 情報科学科
[2] お茶の水女子大学大学院 人間文化創成科学研究科 理学専攻 情報科学コース
[3] お茶の水女子大学 シミュレーション科学教育研究センター
[4] 独立行政法人科学技術振興機構, CREST
[5] 産業技術総合研究所 (AIST) 人工知能センター

{g1220517[a], nakamura.ayako[b], bekki[d]}@is.ocha.ac.jp
mineshima.koji@ocha.ac.jp[c]

1　はじめに：現象説明

述語が選択できる主語や目的語にはある程度制約がある。たとえば、自動詞 “cry” は主語に有生の存在物を指す名詞を要求し、他動詞 “marry” は主語と目的語の両方に人間を指す名詞を要求する。これは一般的に述語の選択制約 (selection restriction) と呼ばれている。本稿では、述語の選択制約の素朴な分析では捉えられない２つの現象に着目する。

まず、自然言語には複数の意味を持つ名詞が存在し、それらは偶然的多義と論理的多義の二種類に分類される (Asher[1])。たとえば、名詞 “bank” は「銀行」と「土手」の２つの意味を持つ。このように、名詞が複数の異なる性質の意味を持ちそれぞれが別の語である場合、偶然的多義という。一方、名詞 “book” は「書かれている情報」と「物理的実体」の２つの意味を持つ。このように、１つの名詞が複数の異なる相の意味を持つ場合、論理的多義という。偶然的多義と論理的多義は、copredication と呼ばれる以下のような構文を許容するか否かによって区別できる。

(1)　a. # The bank specialized in IPOs and is steep and muddy.
　　b. John memorized and burned the book.

(1a) において “specialized in IPOs” は主語が「銀行」であることを要求し、“is steep and muddy” は主語が「土手」であることを要求する。一方、(1b) においては “memorize” は目的語が「情報」であることを要求し、“burned” は目的語が「物理的実体」であることを要求する。このように、等位接続された２つ以上の述語が同じ項に対して異なる選択制約を持つ構文を copredication という。ここで (1a) は偶然的多義、(1b) は論理的多義の例であり、(1a) は copredication を許容しないが、(1b) は許容する。しかし述語に選択制約があるのであれば、２つの述語が項に求めている名詞の性質がそれぞれ異なるにもかかわらず１つの名詞を共有している copredication は一様に不適切であるはずである。よって (1b) のみが許容されるという現象は説明を要する。

次に、「喫茶店でオムレツを食べた客が一人いた」という文脈が与えられれば、以下の (2) を “The man who ate the omelet laughed.” の意味に取ることができる。

(2)　The omelet laughed.

このとき、述語に選択制約があるのであれば、動詞 “laugh” が主語に取ることができるのは有生の名詞のみのはずである。しかし実際は、(2) のように無生の名詞 “omelet” を取ることができる。このような通常ならば選択制約に違反する文が文脈による意味の「強制」が起こることで容認できるようになる現象をコアーション現象といい、これも説明が求められる。

本研究では、これらの現象を分析する枠組みとして、統語論として CCG[6] を、意味理論として依存型意味論 [2] を用いる。述語の選択制約と名詞の論理的多義についての知識は、オントロジーの概念を用いて記述することができる。また、論理的多義性を持つ名詞についての copredication やコアーション構文が容認可能であるということは、述語と項の間に何かしらの意味の変換があるということを示唆している。そこで、オントロジーによって記述された知識に基づいて依存型意味論に動詞と名詞間の意味変換を行う演算子を導入することで、論理的多義やコアーション現象がもたらす問題点の解決を試みる。

2　依存型意味論 (DTS)

依存型意味論 (Dependent Type Semantics) は、依存型理論 [4] に基づいた証明論的意味論である。DTS による意味表示は Π 型と Σ 型が重要な役割を果たす。Π、Σ はそれぞれ自然言語の全称量化、存在量化に対応しており、これらを用いることで先行する文脈を考慮した意味表示を記述することが可能となる。たとえば、(3a) の文の意味表示は DTS では (3b) のように与えられる。

(3)　a. Every man entered.
　　b. $\left(u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{man}(x) \end{array} \right] \right) \rightarrow \mathbf{enter}(\pi_1(u))$

“every” は全称量化として扱われるため、文全体の意味表示は Π 型となる。u は型 **entity** である項 x

と、x に依存した型 $\mathbf{man}(x)$ からなる Σ 型をもち、$\mathbf{entered}(\pi_1(u))$ の $\pi_1 u$ は u の第一要素である **entity** を指す。

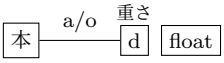
DTS には聞き手にとって未知の意味表示 (underspecified representation) を表す演算子 @ が用意されている [8]。これにより照応を扱うことができ、文脈によって解釈が変わるコアーション現象の分析に @ を含んだ意味表示を用いることが可能である。

3 形式オントロジーと DTS

3.1 オントロジーの構築

ここでは、述語の選択制約と論理的多義の分析のためにオントロジーを導入する。溝口ら [7] によると、オントロジーとは、共通語彙 (概念) を提供する体系的な辞書であり、概念と概念間の関係（意味リンク）を用いて記述される。概念間の関係を明示的に記述することによって、そのバックグラウンドにある暗黙的な情報を明示することが可能となる。人工言語ベースの形式的なオントロジーは、高い形式性をもち、従来の上位・下位概念に基づく概念関係の記述を超えたより精緻な知識の構造的記述が可能になるという利点がある。ここでは、is-a リンク、attribute-of リンクの 2 種類の意味リンクを取り上げる。例えば「自動車」は、「四輪車」や「二輪車」等の異なる概念を含んでおり、これは is-a リンクを用いて記述される。is-a リンクの元を上位概念 (スーパークラス)、is-a リンクの先を下位概念 (サブクラス) と呼ぶ。

各概念がもっている性質は、attribute-of リンクを用いて記述される。例えば、「本は float 型の重さ d を持つ」という知識は以下のように表現される。



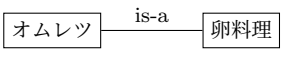
float の位置に現れる概念を「クラス制約」と呼ぶ。上位概念が持つクラス制約は、下位概念に自動的に引き継がれる。

「自動車」「四輪車」「二輪車」「人間」など、コンテキストや他の概念に依存せず、それ自身の性質に基づいて規定される概念は基本概念と呼ばれる。一方、コンテキストに依存して決定される役割を表す概念は、ロール概念と呼ばれる。例えば、「乗客」「運転手」は、ロール概念であり、「自動車」という概念が定めるコンテキストの中での役割を表す。ロール概念が規定する役割を担う基本概念は、クラス制約によって指定される。

3.2 形式オントロジーから DTS への変換

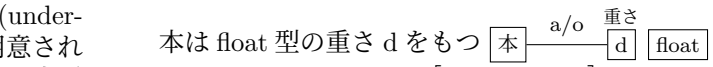
構築したオントロジーを文の意味表示とあわせて推論に使うために、意味リンクの種類に応じて DTS の公理へ変換する必要がある。オントロジーの 2 種類の意味リンクは、それぞれ以下のように DTS の公理へ変換される [5]。

1. **is-a リンク**

オムレツは卵料理である 

$(x : \mathbf{entity}) \to (\text{オムレツ}\ (x) \to \text{卵料理}\ (x))$

2. **attribute-of リンク**

本は float 型の重さ d をもつ 

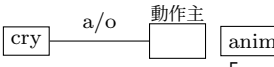
$(x : \mathbf{entity}) \to (\mathbf{book}(x) \to \begin{bmatrix} d{:}\mathbf{float} \\ \mathbf{weight}(x) = d \end{bmatrix})$

述語の選択制約と論理的多義の分析に必要な知識は、特に attribute-of リンクを用いて記述される。

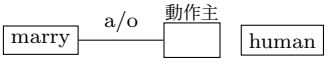
3.3 イベントオントロジー

述語の選択制約の分析のためにイベントオントロジー [3] を構築する。たとえば、“cry” という行為が動作主として “animate” を持つことは、形式オントロジーの attribute-of リンクを用いて記述することができる。「動作主」「対象」などの意味役割はオントロジーのロール概念に相当する。

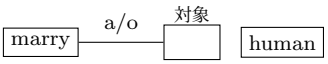
attribute-of リンクのクラス制約を利用することによって、述語の選択制約が表現できる。動作主だけを持つ述語の場合は 1 の形、動作主と対象の両方を持つ述語の場合は 2, 3 の形で DTS の公理へと変換される。

1. 

$(x : \mathbf{entity}) \to (\mathbf{cry}(x) \to \begin{bmatrix} y{:}\mathbf{entity} \\ \mathbf{animate}(y) \\ y =_{\mathbf{entity}} x \end{bmatrix})$

2. 

$(x : \mathbf{entity}) \to (y : \mathbf{entity}) \to (\mathbf{marry}(x, y) \to \begin{bmatrix} z{:}\mathbf{entity} \\ \mathbf{human}(z) \\ z =_{\mathbf{entity}} x \end{bmatrix})$

3. 

$(x : \mathbf{entity}) \to (y : \mathbf{entity}) \to (\mathbf{marry}(x, y) \to \begin{bmatrix} z{:}\mathbf{entity} \\ \mathbf{human}(z) \\ z =_{\mathbf{entity}} y \end{bmatrix})$

4 依存型意味論への各現象の導入とコアーション演算子

4.1 述語の選択制約の導入

本研究では述語の選択制約を各述語のイベントオントロジーにおける attribute-of リンクとして導入する。これにより述語が求める項の性質を指定する関数を合成することができる。たとえば動詞 “laugh”, “memorize”, “burn”, “eat” の関数は、それぞれ図 1 のように得られる。関数 l は自動詞 “laugh” が主語の項に **animate** の性質を持つ名詞を要求していることを表し、関数 $m1$, $m2$ は他動詞 “memorize” が主語と目的語の項にそれぞれ **animate**, **info** の性質を持つ名詞を求めていることを表す。

$$l : (x : \mathbf{entity}) \rightarrow (\mathbf{laugh}(x) \rightarrow \begin{bmatrix} y : \mathbf{entity} \\ \mathbf{animate}(y) \\ y =_{\mathbf{entity}} x \end{bmatrix})$$

$$m1 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{memorize}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{animate}(z) \\ z =_{\mathbf{entity}} x \end{bmatrix})$$

$$m2 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{memorize}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{info}(z) \\ z =_{\mathbf{entity}} y \end{bmatrix})$$

$$b1 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{burn}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{animate}(z) \\ z =_{\mathbf{entity}} x \end{bmatrix})$$

$$b2 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{burn}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{phyobj}(z) \\ z =_{\mathbf{entity}} y \end{bmatrix})$$

$$e1 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{eat}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{animate}(z) \\ z =_{\mathbf{entity}} x \end{bmatrix})$$

$$e2 : (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (\mathbf{eat}(x, y) \rightarrow \begin{bmatrix} z : \mathbf{entity} \\ \mathbf{food}(z) \\ z =_{\mathbf{entity}} y \end{bmatrix})$$

図 1: 述語の選択制約を表す関数

4.2 論理的多義の導入

名詞の論理的多義性は、オントロジーに名詞と各性質の attribute-of リンクを追加することで記述される。たとえば、名詞 “book” に対しては図 2 のような関数がオントロジーより得られる。関数 $book1$ は、型が **entity** である項 x が **book** という性質を持つならば、x は **info** という性質を持つことを意味している。

$$book1 : (x : \mathbf{entity}) \rightarrow (\mathbf{book}(x) \rightarrow \begin{bmatrix} y : \mathbf{entity} \\ \mathbf{info}(y) \\ y =_{\mathbf{entity}} x \end{bmatrix})$$

$$book2 : (x : \mathbf{entity}) \rightarrow (\mathbf{book}(x) \rightarrow \begin{bmatrix} y : \mathbf{entity} \\ \mathbf{phyobj}(y) \\ y =_{\mathbf{entity}} x \end{bmatrix})$$

図 2: 名詞の論理的多義性を表す関数

4.3 コアーション演算子

述語と項の間での意味変換を行うために、まず意味変換後の項を特定する条件について考察する。第一の条件は、意味変換後の項が述語の選択制約により求められている性質を持つことである。たとえば、動詞 “cry” と主語の間に意味変換があるとすれば、意味変換後の項は有生の存在物である必要がある。第二の条件は、意味変換前の項と変換後の項の間には何かしら関係 R が存在し、更に意味変換前の項とその関係 R にある項はただひとつだけであるという条件である。すなわち意味変換前の項と変換後の項の間に何の関係もない場合や、意味変換前の項と関係 R にある項が複数ある場合は意味変換は生じない。

この 2 つの条件を形式化するために、コアーション演算子（図 3、図 4）を導入する。これを動詞の右側から関数適用することで、自動詞の場合には P に動詞、x に主語である名詞句、他動詞の場合には更に y に目的語である名詞句が与えられる。この演算子によって、Pz もしくは Pzx で述語の選択制約により z の持つ性質に制約がかかり、さらに @ に付加された関係 R を文脈から探し出すことによりコアーション後の項 z を特定することができる。

$$\frac{\epsilon}{(S\backslash NP)\backslash(S\backslash NP)} : \lambda P.\lambda x.\begin{bmatrix} z : \mathbf{entity} \\ \left(\pi_1@ : \begin{bmatrix} R : e \rightarrow e \rightarrow type \\ (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (z : \mathbf{entity}) \rightarrow \begin{bmatrix} Rxy \\ Rxz \end{bmatrix} \rightarrow y = z \end{bmatrix}\right) xz \\ Pz \end{bmatrix}$$

図 3: コアーション演算子（自動詞用）

$$\frac{\epsilon}{(S\backslash NP/NP)\backslash(S\backslash NP/NP)} : \lambda P.\lambda y.\lambda x.\begin{bmatrix} z : \mathbf{entity} \\ \left(\pi_1@ : \begin{bmatrix} R : e \rightarrow e \rightarrow e \rightarrow type \\ (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (w : \mathbf{entity}) \rightarrow (z : \mathbf{entity}) \rightarrow \begin{bmatrix} Rxyw \\ Rxyz \end{bmatrix} \rightarrow w = z \end{bmatrix}\right) xyz \\ Pzx \end{bmatrix}$$

図 4: コアーション演算子（他動詞用）

5 分析

5.1 論理的多義の問題

“memorized the book” は他動詞用のコアーション演算子（図 4）を用いると図 5 のように合成される。

$$\dfrac{\dfrac{\dfrac{\text{memorized}}{\begin{array}{l}S\backslash NP/NP \\ : \lambda y.\lambda x.\mathbf{memorize}(x, y)\end{array}} \quad \dfrac{\epsilon}{(S\backslash NP/NP)\backslash(S\backslash NP/NP)} : \lambda P.\lambda y.\lambda x.\begin{bmatrix} z : \mathbf{entity} \\ \left(\pi_1@ : \begin{bmatrix} R : e \rightarrow e \rightarrow e \rightarrow type \\ (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (w : \mathbf{entity}) \rightarrow (z : \mathbf{entity}) \rightarrow \begin{bmatrix} Rxyw \\ Rxyz \end{bmatrix} \rightarrow w = z \end{bmatrix}\right) xyz \\ Pzx \end{bmatrix}}{S\backslash NP/NP} < \dfrac{\text{the book}}{\begin{array}{l}NP \\ : b\end{array}}}{S\backslash NP} > : \lambda y.\lambda x.\begin{bmatrix} z : \mathbf{entity} \\ \left(\pi_1@ : \begin{bmatrix} R : e \rightarrow e \rightarrow e \rightarrow type \\ (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (w : \mathbf{entity}) \rightarrow (z : \mathbf{entity}) \rightarrow \begin{bmatrix} Rxyw \\ Rxyz \end{bmatrix} \rightarrow w = z \end{bmatrix}\right) xyz \\ \mathbf{memorize}(x, z) \end{bmatrix}}{S\backslash NP} > : \lambda x.\begin{bmatrix} z : \mathbf{entity} \\ \left(\pi_1@ : \begin{bmatrix} R : e \rightarrow e \rightarrow e \rightarrow type \\ (x : \mathbf{entity}) \rightarrow (y : \mathbf{entity}) \rightarrow (w : \mathbf{entity}) \rightarrow (z : \mathbf{entity}) \rightarrow \begin{bmatrix} Rxyw \\ Rxyz \end{bmatrix} \rightarrow w = z \end{bmatrix}\right) zbz \\ \mathbf{memorize}(x, z) \end{bmatrix}$$

図 5: “memorized the book” の意味合成

$\mathbf{memorize}(x, z)$ と図 1 における動詞 “memorize” の目的語に対する選択制約を表す関数 $m2$ により、z は性質 **info** を持つという制約がかかる。これと図 2 における名詞 “book” の論理的多義関数 $book1$ を用いることで、関係 R は以下のように合成される。

$$\lambda x.\lambda y.\lambda z.\begin{bmatrix} u : \mathbf{book}(y) \\ \pi_1 book1(y, u) =_{\mathbf{entity}} z \end{bmatrix}$$

従って @ を具体的な関係 R で埋めることにより、文 “John memorized the book.” は図 6 のような意味表示となり、 “John memorized infomation of the book.” の意味に解釈できるようになる。

$$\begin{bmatrix} z : \mathbf{entity} \\ \begin{bmatrix} u : \mathbf{book}(b) \\ \pi_1 book1(b, u) =_{\mathbf{entity}} z \end{bmatrix} \\ \mathbf{memorize}(j, z) \end{bmatrix}$$

図 6: “John memorized the book.” の意味表示

“burned the book” においても同様にして関係 R が以下のように合成される。

$$\lambda x.\lambda y.\lambda z.\begin{bmatrix} u : \mathbf{book}(y) \\ \pi_1 book2(y, u) =_{\mathbf{entity}} z \end{bmatrix}$$

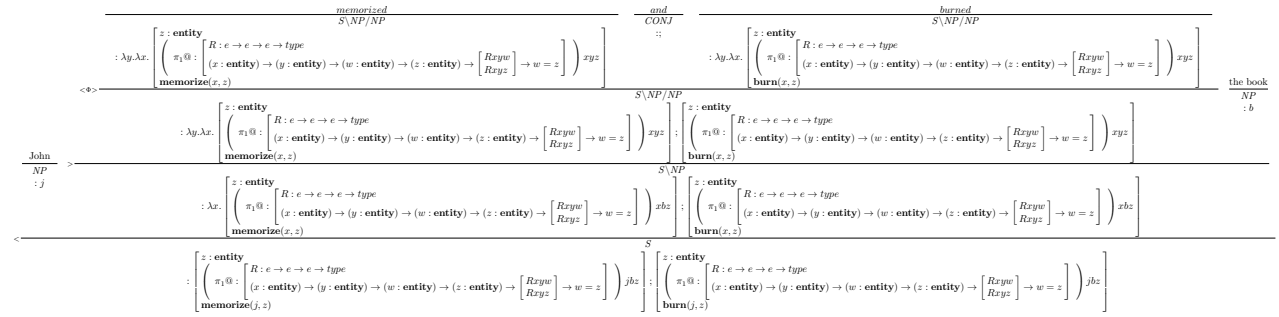


図 7: “John memorized and burned the book.” の意味合成

よって copredication を含む文 “John memorized and burned the book.” は、述語と項の間の意味変換が “memorized” と “burned” のそれぞれで行われるために、図 7 のように意味合成できる。

5.2 コアーション現象

文 “The omelet laughed.” は自動詞用のコアーション演算子（図 3）を用いることで図 8 のように意味合成することが可能である。

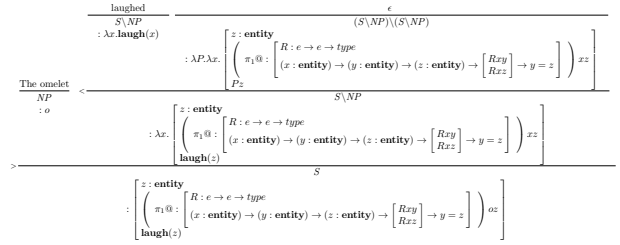


図 8: “The omelet laughed.” の意味合成

laugh(z) と図 1 における動詞 “laugh” の主語に対する選択制約を表す関数 l により、z は性質 **animate** を持つという制約がかかる。これと図 2 における動詞 “eat” の主語に対する選択制約を表す関数 $e1$ を用いることで、関係 R は以下のように合成される。

$$\lambda x.\lambda y.\lambda z.\begin{bmatrix} u : \mathbf{eat}(x,y) \\ \pi_1 e1(x,y,u) =_{\mathbf{entity}} z \end{bmatrix}$$

したがって @ を具体的な関係 R で埋めることにより、文 “The omelet laughed.” は “The man who ate omelet laughed.” の意味に解釈できるようになる。

6 まとめと今後の課題

本論文では、述語の選択制約と名詞の論理的多義の２つの概念と、述語と項間の意味変換を行うコアーション演算子を依存型意味論に導入し、論理的多義やコアーション現象の問題点であった一部の文の意味合成を示した。

現状では、「強制」が起こるにもかかわらず意味合成を行うことができない文脈があること、また、適切でない意味に合成する過剰生成が発生しうることなど、問題点が残っている。これらの問題は、コアーション演算子の調整や追加演算子の導入等により解決可能であると考えられる。また、コアーション演算子を適用するための判断基準が現状では存在していないので、これをどのように導入していくかが今後の課題である。

参考文献

[1] Nicholas Asher. *Lexical meaning in context: A web of words*. Cambridge University Press, 2011.

[2] Daisuke Bekki. Representing anaphora with dependent types. In *Logical Aspects of Computational Linguistics*, pp. 14–29. Springer, 2014.

[3] Ai Kawazoe, Yusuke Miyao, Takuya Matsuzaki, Hikaru Yokono, and Noriko Arai. World history ontology for reasoning truth/falsehood of sentences: Event classification to fill in the gaps between knowledge resources and natural language texts. In Yukiko Nakano, Ken Satoh, and Daisuke Bekki, editors, *New Frontiers in Artificial Intelligence*, Vol. 8417 of *Lecture Notes in Computer Science*, pp. 42–50. Springer, 2014.

[4] Per Martin-Löf. Intuitionistic type theory. *Naples: Bibliopolis*, 1984.

[5] Ayako Nakamura, Koji Mineshima, and Daisuke Bekki. Towards modeling natural language inferences with part-whole relations using formal ontology and lexical semantics. In *CEUR Workshop Proceedings*, Vol. 1517, 2015. 6 pages.

[6] Mark J. Steedman. *Surface Structure and Interpretation*. The MIT Press, Cambridge, 1996.

[7] 溝口理一郎. オントロジー工学. オーム社, 2005.

[8] 佐藤未歩, 戸次大介. 依存型意味論における型推論の定式化と実装. 言語処理学会第 21 回年次大会発表論文集, pp. 461–464, 2015.