Formal Cultural Ontologyとしての「語義」とその関数プログラミングによる実装(2)：様々な型変換

緒方　典裕
大阪大学大学院　言語文化研究科　言語情報科学講座

1　序

言語を記述・比較する場合、古典的なSwadeshの語彙リストのように、「語義は言語間で斉一であることが前提であるが、「語義」はその言語の使用者が属するコミュニティの文化において「概念化(conceptualization)」が行われており、この「コミュニティにおける概念化」を明らかにしなければ「意味」を形式的に扱えないことは、知識工学で「オントロジー」という分野が成立した起因でもある。本発表では、Moravscik [6]のaitiational scheme, Pustejovsky [7]のqualia structureを一般化したデータ型を「qualia」というレコード型として型理論に基づいた関数プログラミング言語OCaml[4]において実装する。この実装の利点として、多くの言語学者が指摘してきたある種の多義性のクラス：Fauconnier [1]が取り扱った「その場限りのmetonymyと定着したmetonyumy」; Pustejovsky [7]で扱われたtype-coercion; culture-specific/trans-culturalな多義性、をレコード型の型推論を基にした型変換によって扱う。

2　レコード型 `qualia`

「語義」は二つの側面を持つといわれる：**外延(extension)**と**内包(intension)**である。Montague [5]の理論に基づく形式意味論では、事実上外延だけを「語義」としてきたが、Moravscik [6]のaitiational scheme、Pustejovsky [7]の「生成語彙」やLakoff [3]を代表とする「認知意味論」では内包に焦点があてられた。関数プログラミング上の型`qualia`はこのような外延と内包の両方をレコード型によって包括したものである。[1]

まず、基本となる型`e`(entity), `year`, `period`, `dimension`, `value`, `rel`(relation), `proposition`, `stage`, `role`, `situation`, `mereologie`(4-dimensional mereology)を以下のように定義する。(以降、「定義」とはOCaml[4]のプログラムによって表し、その内容はfragmentである)

プログラム1

```
type e = Nil|George_Bush
|Alexander_the_Great|Plato
|Greek|Person|Ent of string
|Part of e * string|Work of string
|Picture of e * int|Copy of e * int
|CD of e * int|Stuff of string;;

type place = Poland|Paris|Greece
|North of place|Within of e;;

type dimension = Area|Height|Depth
|Shape|Intelligence;;

type value = Meters of int
|DegreeC of int|Square|High|Low;;

type situation = S of int;;
```

[1]概念を外延の集合と内包の集合の対とするのは、「Formal Conceptual Analysis」[2]がある。

プログラム 1(続き)

```
type year = BC of int| AD of int;;

type period = Around of year
|During of year * year;;

type rel = IV of string * e
| TV of string * e * e;;

type proposition = Gen of rel
| GEN of rel * rel
| Past of rel|At of year * rel;;

type stage =(period list)*(place list);;

type role = Agent|Theme;;

type mereologie =
(stage * (e list) * rel * role) list;;
```

次に型 extension, correlate, qualia をレコード型として定義する：

プログラム 2

```
type extension
= {referent:e; index:stage};;

type correlate
= {activities:proposition list;
disappearance:proposition list;
effects:proposition list;
functions:proposition list;
genesis:proposition list;
quality:(dimension * value) list;
results:e list};;

type qualia = {extent:extension;
correlates:correlate;
mereology:mereologie;
synonymy:string list; taxonomy:e list};;
```

3 Nunberg-Fauconnier 型変換

Fauconnier [1] が示したように、(1a) は (1b) を、(2a) は (2b) を、文脈によっては表しうる：

(1) a. Plato is on the top shelf.

b. Some book by Plato is on the top shelf.

(2) a. Chopin is on the top shelf.

b. Some CD of piano pieces by Chopin is on the top shelf.

しかし、すべての固有名に関して、このようなことが成立するわけではなく、固有名が何かを「work」を残しており、そのコピーが棚にのるものである場合である。つまり、基本的なアイデアとしては、固有名があらわす実体 n を qualia に関数 q によって変換し、そのレコード $q(n)$ とある状況 s, その状況における棚の上にあるものの集合 $X(s)$ に対して

$$q(n).works \cap products2works(X(s)) \neq \emptyset$$

ならば、その文は真であると形式化できる。ただし、$products2works$ は work のコピーを work に変換する関数である。work は型 qualia においては correlate の result に記述される。しかし、棚の上に George Bush の写真があって、その状況を次のように言うことも、少なくとも日本語ではありうる：

(3) ブッシュが一番上の棚にある。

この場合、George Bush に関係するものであれば何でも成立しうるが、状況がそれを支持しなければならない。つまり、固有名のあらわす実体 n とクラス c に含まれるある個体に関係する物の集合からその個体の集合への関数 $relative$ が与えられた場合、

$$n \in relative(X(s))(c)$$

として形式化できる。以上の、(1), (2) のような規則化可能な型変換と、「その場限り」の (3) のような型変換を合わせて、次のようにこれらの例を扱えるよ

うに定義する (ここでは関係のある情報だけを定義する)：

プログラム 3

```
let plato = {extent = {referent = Plato;
 index = [([(During ((BC 427),(BC 347)))],
    [Greece])]};
 mereology = []; synonymy = [];
 taxonomy = [Person];
 correlates = {
  functions =
   [(Past (TV("teach",Plato,
               Alexander_the_Great)))];
  activities =
    [(Past(TV("write", Plato,
             Work("The Republic"))))];
 genesis = [(Past(IV("be born",Plato)))];
 effects = [];
 results = [Work("The Republic")];
 disappearance
  = [(Past (IV("die",Plato)))];
 quality = [(Intelligence,High)]}};;

let q x =  match x with
  Plato -> plato
  | _ -> nil;;

let rec products2works x =
  match x with
  (Copy(y,z)) -> y
 |(CD(y,z)) -> y
 | _ -> Nil;;
```

プログラム 3(続き)

```
let rec relative x =
  match x with
  (Copy(y,z)) -> (relative y)
 |(CD(y,z)) -> (relative y)
 |(Picture(y,z)) -> (relative y)
 |(Part(y,z)) -> (relative y)
 |(Work(y)) -> (Work(y))
 |Plato -> Plato
 |George_Bush -> George_Bush
 | _ -> Nil;;

let is y x (s:situation) =
  if (List.mem x (y(s)))
  then true else if (intersect
  (List.map products2works (y(s)))
       ((q(x)).correlates.results))
  then true else
  if (List.mem (Ent "puttable on")
        ((q(x)).taxonomy)) then
  List.mem x (List.map relative (y(s)))
  else
  (List.mem x (rel0 x
      ((q(x)).correlates.genesis)));;

let on_the_top_shelf s =
 match s with
S(1) -> [(Copy(Work("The Sophist"),12))]
|S(2) -> [(Picture(George_Bush,31))]
|_ -> [];;
```

ただし nil は referent が Nil である以外は何も指定されていない qualia、intersect はリストの交わりが空でなければ true を返す関数、rel0 は「The mushroom omelet left」の「The mushroom omelet」からそれを注文した人に変換するような、「エピソード」に関与した型変換をする関数とする。(紙面の都合上省略する。) すると、つぎの実行例のように (1a) と (3) の論理形式はそれぞれ、状況 S(1),S(2) で true を返す：

実行例 1

```
val is : (situation -> e list) ->
  e -> situation -> bool = <fun>
val on_the_top_shelf :
   situation -> e list = <fun>
# is on_the_top_shelf Plato (S(1));;
- : bool = true
# is on_the_top_shelf George_Bush (S(2));;
- : bool = true
```

4 Pustejovsky 型変換

次に Pustejovsky [7] が示した型変換を扱う。

(4) a. Fitzgerald began a novel.

b. Fitsgerald wants a cappucino.

は

(5) a. Fitzgerald began to read a novel.

b. Fitzgerald wants to drink a cappucino.

と解釈されるというものである。しかし、Pustejovsky [7] の主張に反し、状況によっては、

(6) a. Fitzgerald began to write a novel.

b. Fitzgerald wants to buy a cappucino.

という解釈も可能である。(5) を固定された型変換とすると、(6) は状況依存の型変換であるといえる。これらはプログラム 3 におけるように関数 `began, wants` が `qualia` の `function, genesis` を呼び出し、その `proposition` のリストの中のある `proposition` が状況で instantiate されていれば真、そうでなければ偽と定義すればよい。(紙面の都合上省略する。)

5 まとめにかえて：文化依存な不均一な型変換

このように「語義」の多様な型変換が `qualia` というレコード型を用いて関数プログラミングで実装可能であるが、Pustejovsky [7] の主張に反し、一般的な型変換の関数を定義することは難しい。例えば、「建物」というクラスに含まれる多様な語義を見ても、可能な型変換は次の表が示すように不均一である：

	建物 (-を建てる)	空間 (-にいる)	人組織 (-は 4(000) 名だ)	人物 (-が歩いている)	財産 (-は 5 億円する)	購買品 (-は 200 円する)
うち	○	○	○	×	○	×
家屋	○	○	×	×	?	×
警察	○	○	○	?	?	×
警察署	○	○	×	×?	○	×
新聞	?	○	×	×?	○	○
新聞社	○	○	×	×?	○	×

これらは各述語 (建てる、いる等) がそれぞれ `qualia` の `mereologie` や `correlate` を個別に呼び出すことで実装でき、× が付くものはそれが呼び出せないように `qualia` を定義すればよいが、一般化することは難しいことを示している。いいかえれば、各 `qualia` の関係は美しい木構造ではなく、いびつなネットワークを構成しているといえる。また、「は 4 畳半だ」や「は 9 階にある」などの述語は時代を含めた文化に依存しているといえる。

参考文献

[1] Gill Fauconnier. *Mental Spaces.* The MIT Press, Cambridge, 1985.

[2] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis.* Springer, Berlin, 1999.

[3] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press, Chicago, 1987.

[4] Xavier Leroy, Damien Doligez, Jacques Garrigue, Didier Rémy, and Jérôme Vouillon. The Objective Caml system, release 3.09: Documentation and user's manual, 2005.

[5] Richard Montague. *Formal Philosophy: Selected Papers of Richard Montague (ed. by Richmond Thomason).* Yale University Press, New Haven, 1974.

[6] J. M. E. Moravcsik. Aitia as generative factor in Aristotile's philisophy. *Dialogue*, 14:622–636, 1975.

[7] James Pustejovsky. *Generative Lexicon.* The MIT Press, Cambridge, 1995.