数理解析研究所講究録
第396巻 1980年 190-198

モンテギュ理論をめぐって

電総研　　渕 一博

はじめに

モンテギュ理論は哲学の分野ですすめられてきた形式意味論の枠組を自然言語にあてはめたものである。この形式意味論はもともと自然言語を意識しその「意味」を把えることを意図していたが、必ずしもそれに成功していなかった。そこで、言語哲学と、言語現象自体を対象とする言語学は、これまで別々の流れをなしていた。その状況を変えたのは、モンテギュ理論の出現であった。

ＰＴＱと略称されるモンテギュ理論は、統語論と意味論の統合したモデルを英語（の一部）に適用することに成功した。それ以降，言語学者の中からも、このような論理的手法に興味をもつ人々が現われてきた。

一方、モンテギュ理論は、情報科学（計算機科学）の立場からも興味深いものである。ＡＩの分野ではじまった言語理

解システムと深い関連をもつ可能性がある。

それとともに、モンテギュらが発展させた内包論理は、プログラムの意味論と密接な関係をもつ。プログラム（言語）の意味論自体，形式意味論的アプローチを採り入れて大きく発展してきた。その「指示意味論」の適用が、人工言語と自然言語に対してほぼ同時期に始まったのは面白い現象である。

言語学の流れ

言語学は、チョムスキ理論を中心にして60年代に大きく展開した。しかし、その主たる関心は統語的な現象の解明にあった。意味部門は、実際上は、統語論との関連で触れられるだけであった。

チョムスキ理論は、句構造文法をベースにしていた。統語的現象を把えるため、変形規則が導入された。60年代半ばに確立された「標準理論」では、「深層構造」が設定され、表層の構造は深層構造の変形によって導かれるとされる。深層構造は、ある種の句構造規則によって生成される。深層構造は、意味解釈規則によって「意味表現」に移される。しかし、その意味表現の構造は明示的でなかった。

70年代に入って、意味の問題が前面に出てくるようになって、生成変形理論はいくつかに分流する。一つは「生成意味論」で他は「解釈意味論」である。

生成意味論は、深層構造≒意味構造ととらえる。そのため、深層構造に従来より多くの情報を持たせることになり、それをある種の論理構造とする方向に進もうとする。

解釈意味論では、意味を変える変形を認め、意味解釈に変形が関与するようにモデルが拡張される。さらにそれを進めたモデルでは、変形の痕跡が表層レベルまで残るようにし、意味解釈は、その表層構造に対してなされる。意味解釈の結果は、論理形式に移されるとする。

これらで認められることは、意味表現を論理形式に近づける方向である。

モンテギュ理論の枠組

論理学から出発するモンテギュ理論（PTQ）では、意味表現として、論理形式そのものが設定される。実際には、それは「内包論理」と呼ばれる論理系である。統語構造としては、「分析木」が設定され、この木から論理式への翻訳規則が与えられる。内包論理には、モデル理論的な意味解釈が与えられている。

このようにして、厳密な論理的な枠組の上に、意味論、統語論を統合した言語モデルが作られる。

この構成法は、初期の深層構造モデル、あるいは生成意味論モデルに近いといえる。ただこれらには、意味構造につい

ての明確な論理構成が欠けていた。この類似性のために、モンテギュ理論に興味を持つ言語学者には、生成意味論派の人が多い。

論理式への翻訳

PTQにおける論理式への翻訳の方式を簡単化した例題で見てみよう。

(1) *Every student walks*

の英文に対応して，論理式

(2) $\forall x[S(x) \rightarrow W(x)]$

が考えられる。これらの間はどのようなプロセスでつながれているのであろうか。ここで「ラムダ抽象化」の方法を適用してみる。

(2)' $\forall x[S(x) \rightarrow W(x)]$

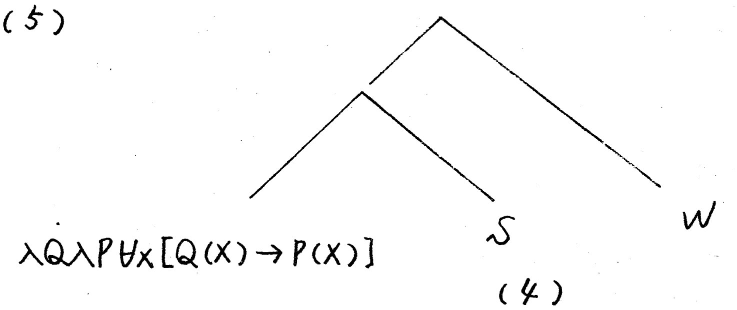
(3) $[\lambda P \forall x[S(x) \rightarrow P(x)]](W)$

(4) $[[\lambda Q \lambda P \forall x[Q(x) \rightarrow P(x)]](S)](W)$

ここで $\lambda Q[---Q---]$ のような形は関数を表わしている。

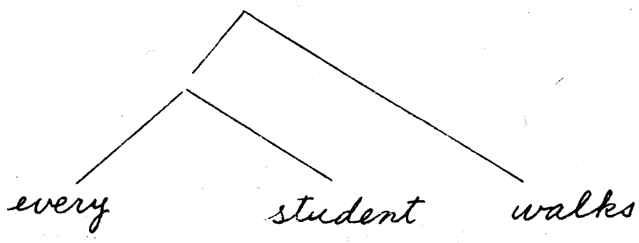
(4)がもつ関数の適用関係を木構造で書けば、

(5)



となろう。一方（1）の英文自身は、

（6）



の統語構造を持つだろう。（5）と（6）を比較すると、意味を表わすと考えられる論理式と表層の英文の構造が同型になっている。この対応によって、

$$every \rightarrow \lambda Q \lambda P \forall x\, [Q(x) \rightarrow P(x)]$$

という every の意味づけができる。このことは、各語に適切に意味（論理式）を附与すると、文の統語構造から、その意味を表わす論理式が合成されるということを示唆する。（4）→（2）への変換は単なるラムダ変換（代入）である。

モンテギュのPTQ理論はこの方式をおしすすめたものである。（2）のような（あるいは（4）のような）論理式を文の意味と考えるのは、このような論理式には形式意味論の考えによる厳密な意味論（モデル的解釈）が与えられているからである。（2）は一階述語論理の形をしているが、モンテギュはそれを拡張した内包論理を用いている。

形式意味論と内包論理

通常の記号論理（一階述語論理）について、その意味論は

モデル理論として確立されている。この考えでは、意味は「指し示すもの」によって決ると考える。すなわち、言語要素は、ある世界(モデル)の中の「もの」と対応づけられる。

このモデルは普通、集合論の中に構成される。そこでは、関数や関係も「もの」として把えることができるからである。

外界との対応(外延)を与えることは、意味を把える一つの道である。自然言語の場合それで十分であろうか。自然言語の文は、絶対的な真偽や指示物をもつとは考えにくい。実際それは文のおかれた環境(状況や文脈)によって異なると考えられる。

このような状況を表わすために様相論理が展開されてきた。様相論理のモデル理論は、ただ一つの世界ではなく、可能世界のあつまりの上で構成される。指示物、真偽は、世界が異なれば異なってくる。

内包論理は、様相論理の一種である。ラムダ形式に「内包化」と「外延化」のオペレータを導入して拡張した、関数形式の様相論理である。

$^{\vee}p$ は、ある(可能)世界が与えられたときの p の値である。$^{\wedge}p$ は、可能世界から値への一種の関数である。(故に、$^{\vee\wedge}p = p$)

LISPとの関連

内包論理はラムダ形式をベースにしており、LISPと非常に近い。オペレータ$^\vee$はEVALに近い。EVALはその実行環境によって異った値をとりうる。$^\wedge$はQUOTEに近い。もし、無変数のラムダ閉式化が許されれば、

$$^\wedge p \sim \lambda(\,)p$$

$$^\vee p \sim p(\,)$$

（この場合も　$[\lambda(\,)p](\,)=p$　）

LISPは *type-free* 的であるが、内包論理は *simple type* 的ラムダ論理である。

<u>形式仕様への変換</u>

ここで、モンテギュ理論的な考えをプログラムの形式仕様に適用する方式を考えてみる。例題として、

find the maximum length of an upsequence contained in $A[0]$ --- $A[n-1]$

をとり上げる。ここで外延的な述語、

upsequence --- $up(u)$

contained in $A[0]$ --- $A[n-1]$ --- $in\text{-}n(u)$

length --- $len(u)=l$

などを導入する。*up*, *in-n*, *len* にはそれぞれが持つ性質が与えられているとする。しかし、これだけでは'*upsequence*'とか'*length*'の意味を形式的に与えたことにはならない。

ところで、

max(n) : the maximum length of an upsequence contained in A[0] --- A[n-1]

とすると、これには、

$$max(n) = k \leftrightarrow$$
$$\exists u . up(u, n, k) \ \& \ \forall j . (\exists u . up(u, n, j) \rightarrow j \leqq k)$$

という形式的な定義が対応する。なおここで、

up(u, n, k) : u is an upsequence of length k contained in A[0] --- A[n-1]

としておく。

このような形式的定義と英語による定義はどのように対応させればよいであろうか。ここで略記法を用いることにして、統語構造を、

$$\underline{max}\,((\underline{len})\,(\underline{of}\,(\underline{a}\,((\underline{up})\ \ \underline{in\text{-}n}))))$$

あるいは標準形式で、

$$\underline{max}\,((\underline{of}\,(\underline{a}\,(\underline{in\text{-}n}\,(\underline{up}))))\,(\underline{len})) \qquad (7)$$

それぞれの単語の意味を、

$$\underline{max} \stackrel{D}{=} \lambda P . \lambda k . (P(k) \ \& \ \forall j . (P(j) \rightarrow j \leqq k))$$
$$\underline{len} \stackrel{D}{=} \lambda l . \lambda u . (len(u) = l)$$

$$\underline{in\text{-}n} \stackrel{D}{=} \lambda P.\lambda u.(P(u) \& in\text{-}n(u))$$

$$\underline{up} \stackrel{D}{=} \lambda u.up(u)$$

とおく。 $\underline{a}$ については、

$$\underline{a} \stackrel{D}{=} \lambda P.\lambda Q.\exists u.(P(u) \& Q(u))$$

これは'a student walks' $\exists x.(S(x) \& W(x))$

などから抽象化されるものである。

$$\underline{of} \stackrel{D}{=} \lambda P.\lambda Q.\lambda x.P(Q(x))$$

これは、Pの属性Qという意味での'Q of P'の意味である。

これらの意味を(7)に代入し、ラムダ変換をほどこしていくと、前に述べた形式的仕様が得られる。ただし、

$$up(u, n, k) = up(u) \& in\text{-}n(u) \& len(u) = k$$

とおく。

ここで面白いのは $\underline{a}$ の働きである。この $\underline{a}$ があることで、形式的定義の中の ∃ が出てくるのである。

ここで述べたのは一例についての試みでしかないが、これは、自然言語による形式的仕様(それからの、論理式による形式的仕様)記述の可能性を示唆すると考えられる。