計算言語学 22－2
（1980. 4. 18）

自然言語のための階層的意味表現システム（LOGICS）とその応用について

西田豊明，堂下修司
（京都大学・工学部）

1．まえがき

計算機を用いた自然言語理解においては，自然言語の「意味」のモデル化と記号化，および意味表現への変換の過程の形式化が重要な問題である。従来，人工知能研究分野では，述語論理，意味ネットワーク，フレーム等の技法を用いて研究がすすめられてきた。しかしこれらはいずれも，実世界（real world）に関する知識の表現形式であり，自然言語表現をこれらの形式に翻訳する過程は人間の力によって試行錯誤的に行なわれ，その定式化は進んでいない。一方，モンテギュの示した文法理論（モンテギュ文法，MG）[2]は，論理学の立場から，統語構造，意味表現，解釈についての統一的枠組を示したものとして興味深い。そこで本稿では，人工知能分野で開発されてきた処理技法をMGに沿って再構成した階層的な意味表現システムを示し，その上で，与えられた自然言語表現の意味解釈を段階的に行なってゆく過程を示す。最後に，この意味表現系の応用例として質問応答システムと簡単な英文和訳システムに触れる。なお，本稿で対象とするのは文脈に著しく依存することのない題材である。また，例題として簡単な英文を用いているがその理由は，応用目的と，英語文法の規範となるすぐれた教科書[11]があったためである。

2．意味表現システム（LOGICS）の構成

本稿で示す意味表現システムは，高階の論理表現言語（Logical Representation Language, LRL），分割された意味ネットワーク（Partitioned Semantic Network, PSN），およびフレームシステム（Frame, FRAME）から成る（図1）。これをLOGICS（Logical meaning representation System）と称する。

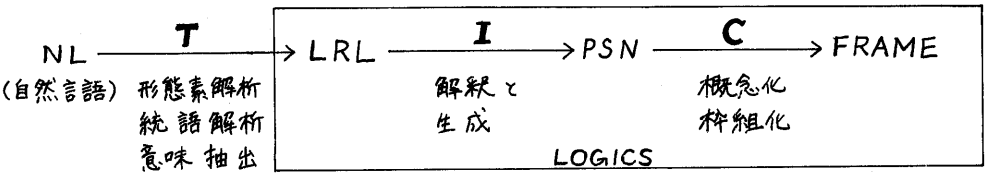


図1．意味表現システムの構成

図1において，自然言語（NL）からLRLへの変換を「翻訳」とよび写像**T**で，LRLからPSNへの変換を「解釈」とよび写像**I**で，PSNからFRAMEへの変換を「概念化」とよび写像**C**で，それぞれ表わすことにする。

2．1．論理表現言語LRLの構文

本稿で用いているLRLはCresswellの著[1]で「λ－深層構造」として述べられている論理的な人工言語に基づいている。LRLの構文はλ－範疇言語で定義される。λ－範疇言語$\mathcal{L}^{\lambda}=\langle F, X, E, \lambda\rangle$は以下のように定義される；

1°統語範疇の集合 Syn：Synは，(1) Nat（自然数集合）$\subseteq$ Syn，(2) $\tau, \sigma_1, \cdots, \sigma_n\rangle \in$ Syn，を満たす最小集合である。

2°記号の集合 F：$F=\bigcup_{\sigma\in Syn} F_\sigma$，ただし$F_\sigma$は範疇$\sigma$に属する記号の有限集合で

あり，$\sigma_1 \neq \sigma_2$ ならば $F\sigma_1 \cap F\sigma_2 = \phi$ 。

3°変数の集合 X： $X = \bigcup_{\sigma \in Syn} X\sigma$， ただし $X\sigma$ は範疇 σ の変数の集合であり，$\sigma_1 \neq \sigma_2$ ならば $X\sigma_1 \cap X\sigma_2 = \phi$ 。しかも，X は F と共通要素をもたないとする。

4°表現の集合 E^λ： $E^\lambda = \bigcup_{\sigma \in Syn} E\sigma^\lambda$，ただし $E\sigma^\lambda$ は次の条件を満足する最小の集合である；(1) $X\sigma \subseteq E\sigma^\lambda$，(2) $F\sigma \subseteq E\sigma^\lambda$，(3) $\delta \in E\langle \tau, \sigma_1, \cdots, \sigma_n \rangle^\lambda$ でかつ，$\alpha_1, \cdots, \alpha_n \in E\sigma_1^\lambda, \cdots, E\sigma_n^\lambda$ ならば表現 $\delta(\alpha_1, \cdots, \alpha_n) \in E\tau^\lambda$，(4) $\beta \in X\sigma$ かつ，$\alpha \in E\tau^\lambda$ ならば，表現 $\lambda\beta[\alpha] \in E\langle \tau, \sigma \rangle$，ただし記号 λ は E のどの要素とも異なる記号であるとする。

2.2. 分割された意味ネットワーク PSN の構成

PSN は LRL の semantics の記述に用いられる。PSN の構成要素は，node，arc，space [3]（以下では paragraph と呼ぶ）から構成される（図2）。

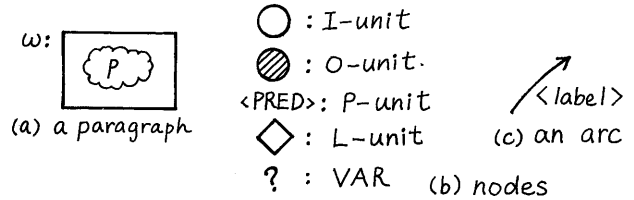


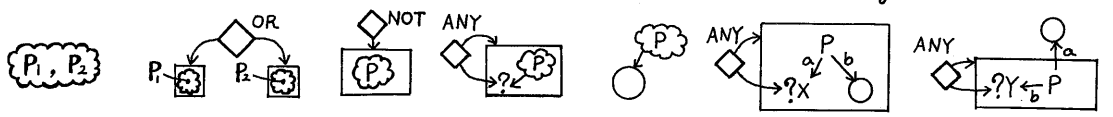
図2 PSN の構成要素（図形表示）

1° paragraph： 一つの可能世界を表現するために用いる。図2(a)の箱は，メタ述語 T(w, p)，「命題 p は可能世界 w で真である。」[4] に対応する。paragraph は命題を引用して個体として参照するために用いる。

2° node：次の5種類の node を用いる；(1) I-unit：個体定数を表わす。(2) O-unit：内包的記述によって個体を表わす。(3) VAR：個体変数を表わす。I-unit，O-unit，および paragraph を代表できる。(4) P-unit：個体間に成り立つ関係を記述する。(5) L-unit：論理接続詞および量限定詞に対応する表現を行なう。本稿では，図2(b) のような図形表示を用いる。

3° arc：実際に node や paragraph を連結して複合表現をつくり出す。各 arc には，識別のためのラベル（格ラベル）を付ける。格ラベルは PSN の意味には影響しない。

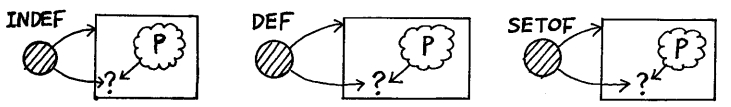
これらの要素を用いたメタ論理式に対応する表現を図3(a)～(g)に示す。†



AND(P_1,P_2) (a)　OR(P_1,P_2) (b)　NOT(P) (c)　∀?X.P(?X) (d)　∃?X.P(?X) (e)　∀?X. ∃?Y.P(?X,?Y) (f)　∃?X. ∀?Y.P(?X,?Y) (g)

図3. メタ論理式に対応する PSN 構造

このように PSN 構造では AND とヨが implicit な表現となる。∀のスコープによるヨの意味の変化は，skolem 定数のおかれる位置によって区別される（図3(f)と(g)）。次に O-unit による内包的記述の例を示す（図4）。INDEF[?X;P(?X)] は $\in x P(x)$



(a) INDEF[?X;P(?X)]　(b) DEF[?X;P(?X)]　(C) SETOF[?X;P(?X)]

図4. O-unit による内包的記述

†) 以下ではネットワーク図式のかわりにそれと等価なメタ述語表現を用いることがある。メタ述語は大文字で書き，object 言語である LRL 表現は小文字で書くことによって区別する。またメタ変数は ? を接頭語として用いることで区別する。

に相当するが∃x P(x) は前提としない。また，DEF[?X; P(?X)] は ix P(x) に相当するが ∃!x P(x) は前提としない。

2.3. PSN構造の探索とパターンマッチング

PSN構造の探索は，内部処理の基本操作である。PSN構造の探索においては，はじめにPSN構造のインデクスから与えられたパターンに適合する可能性のある候補をあげ，次に，各候補が実際にパターンに適合しているかどうかを詳細に調べる。これらの処理はparagraphを単位として行なう。すなわち，探索においては，与えられたパターンを含むparagraphに適合するparagraphを探索するという問題におきかえられ，また適合の検査は与えられた二つのparagraphが適合するか否かという問題が出発点となる。paragraph P_1 が P_2 に適合するということは，P_2 に含まれている命題の連言が，P_1 に含まれている命題を含意すること，と意味付けできる。

1° 手続き find(pattern): patternに適合するparagraphの探索。

(step 1) 作業用のparagraph(KP, key paragraph)を生成し，その中にpatternを解釈したPSN構造を生成する。

(step 2) PSN構造のインデクスを調べ，KPに適合する可能性のあるparagraph(CP, candidate paragraph)のリストを作る。これには，KPに含まれる述語を含んだparagraphを探せばよい。ただし，作業領域となったparagraphは除外する。

(step 3) CPのリストに含まれている各CPに対してmatch(KP, CP)を実行する。手続きmatchは，もしKPとCPが適合していれば，KPとCPの間の対応関係を示す連想リストを結果として返してくる。この連想リストにCPの識別番号をつけたものが集められて検索結果となる。手続きmatchがKPとCPの間の適合関係を示すことができなければ，空リストNILがmatchの値となり，そのCPはfindの値から除外される。

2° 手続き match(KP, CP): paragraph KPがCPに適合するか否かの検証。

(step 1) KPの全命題のリストをLとする。結果変数r，(初期値=NIL)。

(step 2) L=NILなら成功，結果を返す。

(step 3) Lから1個の要素をとり出し，これをlとする。lがCPの全命題の連言に含意されているかどうかを判定する手続き，p-implies を呼ぶ。p-implies(CP, l)がnon-NILであれば，これはlがCP中の命題の連言に含意されていることが示せた訳で，この結果をrに加え(step 2)へ，さもなければmatchは失敗で，return(NIL)。

3° 手続き p-implies(CP, l): 命題lをparagraph CPが含意するか否かの検証。この検証には，PSNの述語の辞書中に含まれている上位下位関係，および，論理接続詞に関する自然演えき法に基づくforward / backward chainingを用いる。引数にparagraphをとるような述語に関しては，その引数が肯定的(positive)か，否定的(negative)かを示すmarkerによって荒いmatchingをとる。この意味は，paragraphが引数である場合を，$(A \rightarrow B) \rightarrow (KNOW(X, A) \rightarrow KNOW(X, B))$ のような場合(positiveな場合)と，$(B \rightarrow A) \rightarrow (INHIBIT(X, A) \rightarrow INHIBIT(X, B))$ のような場合(negativeな場合)に分けて扱うという事である。手続きp-impliesは;

(step 1) CPの全命題のリストをMとする。lがMに含まれていれば成功。

(step 2) (特別な場合のチェック) lのgovernerの辞書記述が特別の取り扱いを指示していればそれを用いる。引数が内包的記述されているときは，その記述間の

含意を検査する。paragraphの引数があればpositive / negativeのmarkerに従って後の過程を制御する。

(step3)(自然演えきに基づく検査)

ℓ = OR(A, B)であれば, match(A, CP) ∨ match(B, CP)。

ℓ = NOT(A)であれば, Mの要素中にm = NOT(B)なるmが含まれかつ, match(B, A)となっているかどうかを調べる。

ℓ = IMPLIES(A, B)であれば, match(NOT(A), CP) ∨ match(B, CP)。

Mの中にANY(?X, P(?X))があれば, p-implies('P(?X)を含むparagraph', ℓ)。

Mの中にIMPLIES(A, B)があってp-implies(B, ℓ)であればmatch(A, CP)を試みる。

2.4. フレームの役割

フレームシステム(FRAME)は, PSN構造のまとまりをとらえて処理するために用いる。すなわち, FRAMEの使用目的としては, LOGICSの処理系の内部でのPSN構造の検索の効率化, ユーザがPSN構造の上での処理を行なうためのプログラミングシステム, common senseに基づくPSN構造のチェックや欠けている情報の補充等の推論の記述, 等があげられる。FRAMEは, 手本となるスキーマフレームとそれが実際のデータに対して例示された例フレームの集まりから成る。それぞれの構造は;

〈スキーマフレーム〉::= (〈名前〉, 〈格スロット〉, 〈変数〉, 〈例示条件〉, 〈格の定義〉, 〈手続き付加〉)

〈例フレーム〉::= (〈スキーマフレーム名〉, 〈値の代入された格スロット〉, 〈変数の束縛リスト〉)

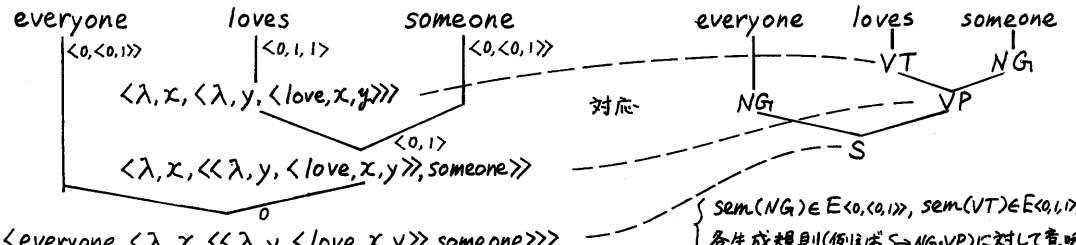
〈例示条件〉には, フレームがどのようなパターンに適用されるかを記述する。このパターンに適合するPSN構造が見い出されるとそれに対して例フレームを生成する。格スロットはフレームのとらえたPSN構造の特徴を属性-属性値対として表示するために用いる。〈格の定義〉は格スロットの表わす属性に対する属性値を, 例示されたPSN構造からどのようにして取り出すかを記述する。〈手続き付加〉の部分は, フレームによる処理の内容を示すもので, 次のようなことを行なう; (1) 格スロットの値のチェック: 格スロットの値が未定義であったり, common senseから見て不自然であれば, 推論や対話を行なって修正する。(2) フレームの内容表示。

3. 自然言語の翻訳と解釈

この章では, 自然言語からLRLへの翻訳Tのあらましと, LRLからPSNへの解釈Iの具体的内容について述べる。

3.1. 自然言語の翻訳Tについて

翻訳の方針として, MGと同様に同一の文法カテゴリには同一範疇のLRL表現を割付けることにする。LRLは2.1.で述べたように文献[1]のλ-深層構造に基づいているが, λ-深層構造から表層構造への生成過程をより制限している。すなわち, [1]で述べられているλ-深層構造では, 関数記号を表現のどこにおいてもよいとした上で, 「λ-範疇言語$\mathcal{L}^{\lambda}$の文αの浅い構造とは, αにあらわれる順に並べられた, αに含まれる固有記号からなる系列S(α)のことである」と定義して自然言語をモデル化している(図5(a))。従って文法規則はいわゆる「意味文法」であって, 従来の文法と全く異なっている。しかし我々は文法カテゴリに従来のもの(規範文法, [11])を用い, その各文法カテゴリにλ-範疇言語表現を対応づけるという方法をとった(図5(b))。



(a) Cresswell のモデル化　　(b) 我々の処理方法

図5. 自然言語文の翻訳T の比較

我々はこのような方法で写像Tをパーザに組み込み，主に英語について実験を行なった。この実験で対象とした文法カテゴリとそれに対するLRLの範疇の対応を表6に示す。本稿はTの具体的アルゴリズムを示すことは目的としていないので，これ以上言及せず，かわりに，付録1に，実験した範囲でどのように各文法カテゴリのLRL表現が生成規則に従って合成されてゆくかを示す。

表6. 簡単な英語の文法カテゴリに対応するLRL表現の範疇

文法カテゴリ	LRLの範疇
文	0
(個体)	1
述部	<0,1>
自動詞	<0,1>
他動詞	<0,1,1>
名詞句	<0,<0,1>>
普通名詞	<0,1>
決定詞	<<0,<0,1>>,<0,1>>
形容詞	<<0,1>,<0,1>>
前置詞	<<0,0>,1>
文副詞	<0,0>

3.2. LRLの解釈Iについて

LRLの解釈Iにおいては，与えられたLRL表現を段階的にたどり，表現の参照の解決等を行ないながら，PSN構造を生成してゆく。解釈Iのインプリメンテーションを手続き gen_{PSN} と書く。gen_{PSN} は，

(<cxt>, <lrl>, <parag>, <alist>)

という引数で呼ばれる。<cxt>は文脈の表現，<lrl>は解釈すべきLRL表現，<parag>は生成されたPSN構造をおくためのparagraphの識別番号，<alist>は変数束縛リストである。gen_{PSN} の返す値の形式は，

(<cxt'>, <red>, <alist'>, <ptr>)

である。<cxt'>は新しい文脈表現，<red>は簡約化された表現，<alist'>は新しい変数束縛リスト，<ptr>は結果として生成されたPSN構造である。(具体的な意味は以下に述べる。)

gen_{PSN} では，与えられたLRL表現の関数部(governer)に関するPSN構造生成のための辞書記述をとり出す。各辞書項目には，TYPE, CASE, PROG という属性に関する記述が与えられている。TYPE 属性は，辞書項目の形式を示すもので，{PRED, PROG} のいずれか一つが選択される。

1° PRED型の辞書項目：通常のfirst order の述語のための項目であって，生成すべきPSN構造のとる格パターンを記述する。この格パターンはCASE 属性の値として次のような形式で与える；

CASE = (<case-slot名>, <extensionality>, <value type>,
<set indication>, <condition>)

ただし，<extensionality> = {EXT: 外延性つまりfillerを外延化する，INT: 内包性}
<value type> = {0: 文引数，つまりparagraph∨VAR，1: 個体の引数，つまりI-unit∨O-unit∨VAR}
<set indicatron> = {ITEM: fillerはItem，SET: fillerはSet}
<condition> = fillerxの満たすべき条件P，P(x)が真にならなければ，error。

PRED 型の辞書項目に対して gen_{PSN} は P-unit は与えられた <parag> 中に生成し，さらに，与えられた <lrl> の引数に対する PSN 構造を生成する。この結果，

（<cxt>, NIL, <alist>, <生成された P-unit の識別番号>）

という値が返される。

2° PROG 型の辞書項目：これは，first order の述語の例外，論理接続詞，高階述語に関する辞書項目である。この形式の項目を処理するためのプログラムが PROG 属性として埋め込まれる。プログラムの形式は，

λ（<文脈>, <args>, <paragraph>, <束縛リスト>）[<関数本体>]

で，関数本体の返す値は，

① （<新しい文脈>, NIL, <新しい束縛リスト>, <生成された PSN 構造>）

② （<新しい文脈>, <λ-式>, <新しい束縛リスト>, NIL）

のいずれかでなければならない。②の場合，<λ-式> はもとの述語の簡約化された（reduced）ものであり，これに再び，与えられた <args> を apply することにより ひきつづき解釈を続行する。

3.3. 文な文法カテゴリに対する LRL 表現の解釈手続き

はじめに主な文法カテゴリに属する語に関する辞書記述を示す（表7）。

表7. 主な文法カテゴリに属する語の辞書記述

品詞	辞書記述
普通名詞	TYPE=PRED, CASE=((SUBJ, EXT, 1, ITEM, φ))
抽象名詞	TYPE=PROG, PROG=叙述に必要な引数を文脈<cxt>より補った後，PSN構造を生成する
属性名詞	TYPE=PROG, PROG=属性の目的語を文脈<cxt>から補って，属性名詞に対応する2引数述語の一方に代入したものを返す．
決定詞(a,an)	TYPE=PROG, PROG=λ?P[?P(INDEF[?X; Q(?X)])]，ただしQは引数<args>の第1番目の要素（名詞相当部分）の解釈。
決定詞(the)	TYPE=PROG, PROG=findobj[INDEF[?X; Q(?X)]]，ただしQは上と同様。発見されればそれをXとすると λ?P[?P(X)]，発見されなければ λ?P[?P(DEF[?X;Q(?X)])]
決定詞(every)	TYPE=PROG, PROG=λ?P[ANY[?X; Q(?X)→?P(?X)]]，Qは上と同様
形容詞	TYPE=PROG, PROG=名詞を名詞に写像するプログラム。引数となる名詞の意味を詳細化する。
is	TYPE=PROG, PROG=二つの引数を等置する手続き，とくに主語の名詞を文脈<cxt>に比較属性としてpush downしておく
自動詞	TYPE=PRED, CASE=((SUBJ, EXT, …, …, …))
外延的他動詞	TYPE=PRED, CASE=((ACTOR, EXT, …, …, …)(OBJ, EXT, …, …, …))
内包的他動詞	TYPE=PRED, CASE=((ACTOR, EXT, …, …, …)(OBJ, INT, …, …, …))
前置詞	TYPE=PROG, PROG=文脈<cxt>に前置詞とその目的語になっている名詞の対をpush downする。これは抽象名詞や属性名詞の解釈，あるいは受動態の解釈（意味上の主語を求める操作）のとき，<cxt>から取り出される。

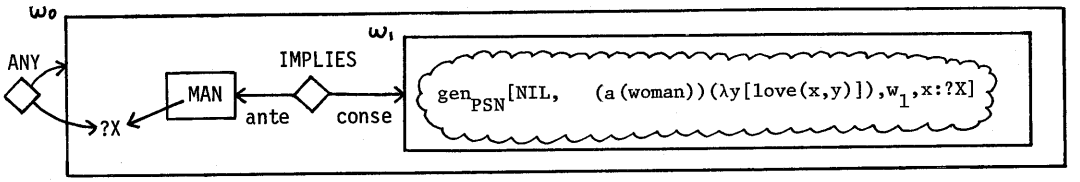
次に全体の概要を示す例をあげる。

（例1）例文 Every man loves a woman. の解釈。LRL表現は，

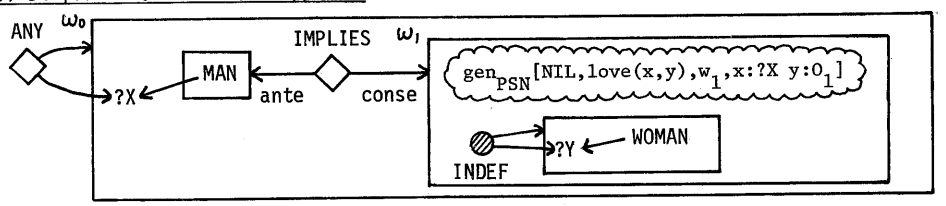
(every (man))(λx[(a (woman))(λy[love(x, y)])]) †

†) この文は2通りの読み方があってあいまいであるとされている（∀∃読みと∃∀読み）。このLRL表現は前者に対応する。

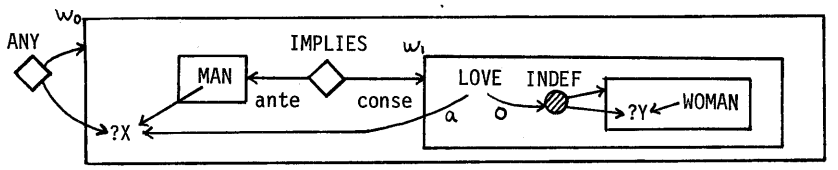
(a) step1　every(man)の解釈。



(b) step2　a(woman)の解釈。



(c) step3　loveの解釈。



(d) step4　loveのobject格のフィラーの外延化

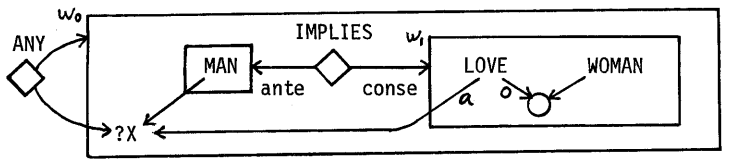


図8．LRL表現 (every(man))(λx[(a(woman))(λy[love(x,y)])]) の解釈の過程

解釈の過程を図8(a)～(d)に示す。はじめに名詞句が解釈されてPSN構造が生成されてゆく。文の意味表現のgovernerである動詞句の意味の解釈は，そのすべての引数が解釈されてPSN構造が生成された後に行なわれる。最後のステップ(d)では，動詞句の格(case)とそれに与えられたfillerとの整合をとる。その結果，内包的表現が外延的表現でおきかえられている。

(例2) 句 the development of the system の解釈。LRL表現は，

the(λy[(the(system))(λx[((xap(of))(x))(development(y))])])

(step1) the の解釈。 findobj[INDEF[?X;[I((the(system)…)_{y←?X}　を実行する。

findobj を実行するために working space として作業用 paragraph w_1 を生成し，その中に I((the(system)…)_{y←?X}) をパターンとして生成する。

(step2) the(system)の解釈。 findobj[INDEF[?Y;SYSTEM(?Y)]] により，作業用 paragraph w_2 が生成される。w_2 の中にはメタ述語 SYSTEM(?Y) に対応するPSN構造が生成され，find[w_2]が実行される。今，この探索が成功したとして?Y に適合する node が Aであったとする。変数xにAを束縛して，

(step3) (((xap(of))(x))(development(y)))_{y←?X, x←A} の解釈。

文脈<cxt>に of：形容詞的：A を push down して，

(step4) (development(y)) $y \leftarrow ?x$ の解釈。文脈 <cxt> から 'development の対象' として of:形容詞的 という属性の値をとり出す (A)。これらを用いて 'development of the system' を近似する PSN構造として図9のような構造を生成する。

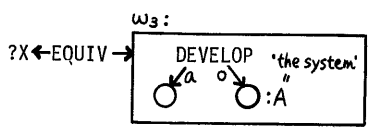


図9. 'the development of the system' の解釈の途中で生成された PSN構造

(step5) findobj の実行。

?X の探索は paragraph w3 (図9) の探索に置きかえられる。もしそのような paragraph が発見されれば、それが最終結果となる。発見されなければ図9の構造を DEF で内包的記述したものが最終結果となる。

次に、付録1にあげていてまだ解釈を示さなかったものについて、解釈の概要を表10に示す。ここでも、文脈 <cxt> を介したメッセージの通信の機構を利用している。

表10. 意味解釈の概要

品詞	解釈
代名詞	人称代名詞 I, you に対しては、文脈 <cxt> から speaker, hearer 属性をそれぞれ取り出し、<0,1> の範疇である引数に代入する。she, he に対しては、それぞれ、the(female), the(male) として探索を行なう。
関係節 [which(文)]	∧?P[∧?X[AND(?P(?X), <文の解釈>)]]。 文の解釈を行なうとき、which は文脈 <cxt> に #ante = ?X を push down しておく。文に含まれる <0,<0,1>> の範疇の語 #ante の解釈時にこの属性をとり出して用いる。
受身の主語 [*en(<他動詞>)]	真の主語 *psubj が示されているときはまず、副詞句として働く *psubj(x)∈E<0,0> の文に真の主語が代入され、次に、文脈 <cxt> 中にこれを push down する。述部 *en(<他動詞>) の解釈のとき、文脈 <cxt> の *psubj 属性を調べる。見つかればそれを、さもなければ不定 node を、<他動詞> の主語として補う。
文の名詞化 [that, whether]	新しく paragraph を生成し、その中に補文に対応する PSN構造を生成する。
複数化	集合記述用の 0-unit SETOF を用いた PSN構造を生成する。
不定詞	新たに paragraph を生成して不定詞の文の部分に相当する PSN構造を生成する。不定詞では通常動詞の主語が省略されているか for 前置詞句で指示されている。指示されている場合は受身の主語同様、文脈 <cxt> を介して通信を行なって補われる。
所有格 [*poss]	名詞句+所有格によって限定されている名詞(相当語)が、普通名詞、抽象名詞の場合、前者の場合 所有を表わす述語 POSSBY で連結し、後者の場合、述語の主格(語)を文脈 <cxt> を介して指示する。
格の整合	動詞の格のパターンと引数として与えられた PSN構造 (filler) が整合するように調整する。整合の規則は次のようである;

extensionality (外延性)

SOURCE \ TARGET	INT	EXT
INT		外延化
EXT	そのまま	

value type (タイプ)

SOURCE \ TARGET	0	1
0		失敗
1	個体を含む paragraph	

set indication (集合性)

SOURCE \ TARGET	ITEM	SET
ITEM		{ITEM}
SET	elementに関する制約	

```
(I WANT HIM TO WRITE A PAPER) ←――― 入力文
"PARSED"

(#DCL (I (LAMBDA X28 (HE (LAMBDA X29 ((INF (#SUBJ (LAMBDA X26 ((A PAPER)   ←―― LRL表現
 (LAMBDA X27 (WRITE X26 X27)))))) (LAMBDA X30 (WANT1 X28 X29 X30)))))))))   (LISP記法)

"GENERATED"

PAGE.1                                                                PAGE.2   ←―― 生成のための構文木

                                                                        +BUN
                                                                          |
                                                  +FBUN-------------------------------------------+END
                                                    |                                               |
                                                  +TBUN                                             |
                                                    |                                               |
                                                  +JUTU                                             |
                                                    |                                               |
          +KAKU-------+KAKU-------------------------------------------+KAKU----------------+DOUKU  |
            |           |                                               |                     |      |
+MEIKU--+JO +MEIKU--+JO                                   +MEIKU-------------+JO  +DOUKU1-+JO      |
  |       |   |       |                                     |                 |     |     |      |
+MEISHI   |  +MEISHI  |                         +TBUN-------------------+FUZOKU  |  +DOUSHI  |      |
  |       |   |       |                           |                     |     |     |     |      |
  |       |   |       |                         +JUTU                   |     |     |     |      |
  |       |   |       |                           |                     |     |     |     |      |
  |       |   |       |          +KAKU------------+KAKU--------+DOUKU   |     |     |     |      |
  |       |   |       |            |                           |        |     |     |     |      |
  |       |   |       |            |      +MEIKU------+JO +DOUKU1-+JO    |     |     |     |      |
  |       |   |       |            |        |           |     |     |    |     |     |     |      |
  |       |   |       |            |  +GENTEI-+MEIKU1   |  +DOUSHI  |    |     |     |     |      |
  |       |   |       |            |        |           |     |     |    |     |     |     |      |
  |       |   |       |            |      +MEISHI       |     |     |    |     |     |     |      |
  |       |   |       |            |        |           |     |     |    |     |     |     |      |
WATASI  WA  KARE    NI    *      *     RONBUN  WO  KAK U    *   KOTO  WO  NOZONDEI RU   *     .

<<<<<  WATASI WA KARE NI RONBUN WO KAK U KOTO WO NOZONDEI RU .  >>>>>   ←―― 訳文
```

図11. LRLを中間言語とした英文和訳の例

4. LOGICSの応用について

LOGICSは文献の案内を行なうシステム[6],[7]で，知識ベース構成のために用いた知識表現を一般化したものである。関係形式についての質問応答は，入力文に関する意味表現がtop levelが命令文，疑問文，平叙文のどれであるかによって処理手続きを切り換えて呼び出して，質問応答やデータの蓄積等を行なう。また日本語についても本稿と同様の方式を適用できることが示されている。

LOGICSの第二の応用例として意味表現を中間言語とした機械翻訳を考えることができる。我々はその第一歩としてLRLを中間言語としたものを構成中である。[9]その一例を図11に示す。

5. まとめ

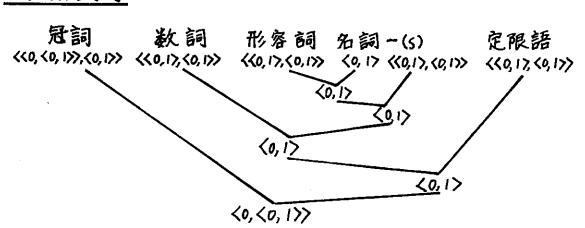
本稿では自然言語のための意味表現システムについて意味解釈を中心に述べた。現在，LOGICSのソフトウェアをLISP上に構成中である。すでに自然言語用のパーザは試作されており（LINGOL-K：日本語用[5][11]，EASY：英語用[8]），現在，翻訳Tのインプリメントである文法規則を作成中であり，本稿で述べた部分は完成している。解釈Iは未だわずかしかインプリメントされていない。従って辞書の規模もパーフォーマンスを調べる程度のものであり，T, Iの基本ルーチンが完成した後，充実させたい。

文献

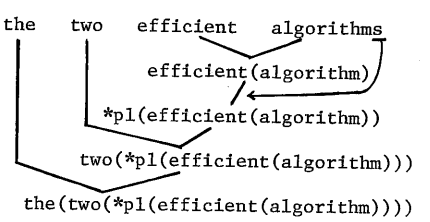
【1】Cresswell著，石本，池谷訳，言語と論理，紀伊國屋書店，'78 【2】Dowty, A Guide to Montague's PTQ, Indiana University Linguistic Club, '78
【3】Fikes, Hendrix, A Network-Based Knowledge Representation and its Natural Deduction System, Proc. IJCAI-77 【4】Moore, Reasoning about Knowledge and Action, Proc. IJCAI-77, 【5】仲田，係り受け解析を中心とした日本語解析システムの構成，京都大学卒論，'79
【6】Nishida, Doshita, The Framework of Knowledge Representation and its Retrieval in LGS, Proc. IJCAI-79, 662-664, 【7】Nishida, Doshita, A Knowledge-based Literature Guide System, IFIP Congress 80 (to appear), 【8】西田，堂下，英文解析用パーザ（EASY）の作成，昭和55年度情報処理全国大会 1198, 【9】西田，浦野，山中，堂下，意味解析に基づく英文和訳について，昭和55年度情報処理全国大会 (to appear), 【10】柳原，日本語の述部解析システムとそれを用いた和文英訳について，京都大学卒論，'80, 【11】Quirk, Greenbaum著，池上訳，現代英文法 大学編，紀伊國屋書店，'77

付録1. 自然言語からLRLへの翻訳例

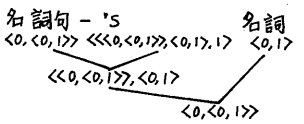
(1) 名詞句



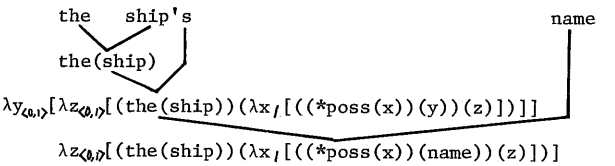
(例)



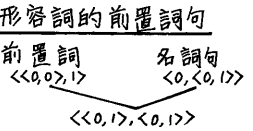
(2) 所有格



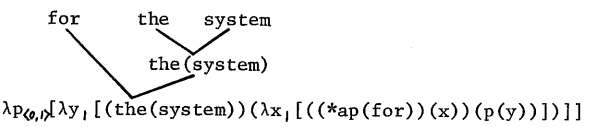
(例)



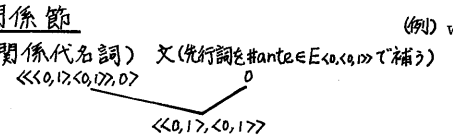
(3) 形容詞的前置詞句



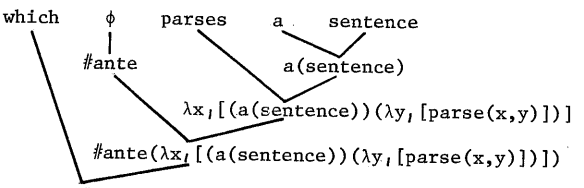
(例)



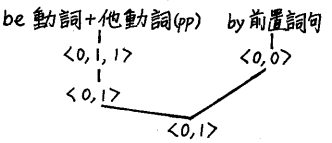
(4) 関係節



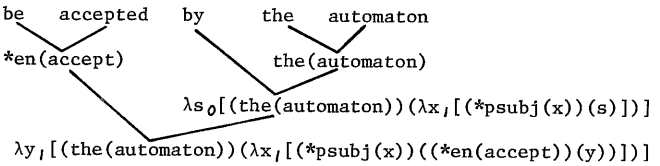
(例)



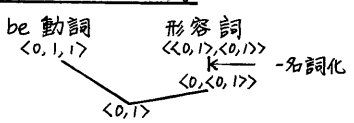
(5) 受け身文



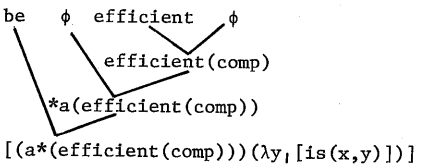
(例)



(6) be 動詞+形容詞



(例)



LRLに含まれる特別な記号一覧

記号	範疇	コメント
*Pl	<<0,1>,<0,1>>	複数化
a*	<<0,<0,1>>,<0,1>>	表層に違しない冠詞
comp	<0,1>	比較属性によって特定される性質
which	<<<0,1>,<0,1>>,0>	関係代名詞
*poss	<<<0,<0,1>>,<0,1>>,1>	所有格の's の表現
*en	<<0,1>,<0,1,1>>	受け身演算

*ap	<<<0,1>,<0,1>>,1>,<<0,0>,1>>	形容詞句形成前置詞
#ante	<0,<0,1>>	先行詞に対応
that	<<0,<0,1>>,0>	that節, 文を名詞類に変換する.
inf	<<0,<0,1>>,<0,1>>	自動詞(相当句)から名詞類をつくる
parti	<<<0,1>,<0,1>>,<0,1>>	自動詞(相当句)から分詞をつくる
whether	<<0,<0,1>>,0>	疑問の名詞句
nom	<<0,<0,1>>,<0,1>>	自動詞(相当句)から動名詞をつくる