

ポータビリティを目指した日本語インターフェースの試作
 松島利幸 小室睦 小野山隆 山本昌彦 小柳和子
 (日立ソフトウェアエンジニアリング(株))

1. はじめに

私たちは、ユーザフレンドリなマンマシンインタフェース実現のための基本技術である自然言語理解をテーマとして取り上げた。本報告では、オブジェクト指向型手法を用いた意味解析処理とkonoNUSという自然言語インタフェース作成システムについて述べる。

2. 基本方針

自然言語インタフェースの意味理解実現について、二つの方針が考えられる。ひとつは、広汎な自然言語を普遍的表現形式に変換するもの。もうひとつは、意味理解の文章範囲をインタフェースの対象になるターゲットシステムに対するものだけに限定してしまうものである。前者は、技術的に実現が難しく、後者の方針を採用した。

つまり、

(1) インタフェースの意味解析処理は、ターゲットシステムに依存しても良い。

(2) 意味解析処理を効率的に記述する方法を開発し、種々のターゲットシステムに対するインタフェースを容易に作成できるようにする。

3. konoNUSの概要

konoNUSは角田、中村が開発したkonoLisp¹⁾を用いて開発された。konoNUSは、現在、68000 CPU メモリ2Mのパソコン上で稼動している。konoNUSの概要を図1に示す。konoNUSでは、意味解析処理をオブジェクト指向型の意味解析言語konoNULを用いて「意味辞書」に記述している。そして、「意味辞書」を交換することにより、種々のターゲットシステムに対応する自然言語インタフェースが実現できる。

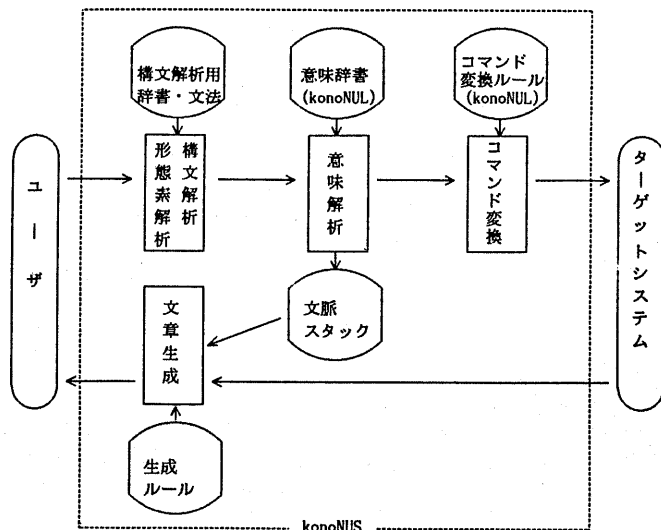


図1 konoNUSの概略構成図

(1) 形態素解析・構文解析モジュール

処理方法はボトムアップであり、文法規則としては次のような文脈規定型文法²⁾を使っている。

$$A\alpha \rightarrow \beta\alpha \quad A \in V_N, \beta \in V^+, \alpha \in V^*$$

また、文節間の係り受けについて複数の可能性がある場合には、近い方の文節にかける。こうすると、意味的に間違った係り受けの構文木が生成されることもあるが、意味解析処理の際に、係り受けが修正される。

(2) 意味解析モジュール

構文木を入力として、意味の標準表現に変換する。この変換処理は、konoNULによって

辞書に記述されている。生成された意味標準形は、文脈スタックに格納されて、文章生成の際に使われる。

(3) コマンド変換モジュール

意味の標準表現から、ターゲットシステムに対するコマンドに変換する。この変換処理も、konoNULによって記述されている。

(4) 文章生成

ターゲットシステムからの結果と、文脈スタックに貯えられた情報から、処理要求に対して的確な返答を、文章、表にして出力する。

4. konoNUSの適用例

ターゲットシステムとして、人や会議室のスケジュールを管理するシステムを例にとり、konoNUSで検索用の自然言語インターフェイスを試作した。その処理例を図2に示す。処理時間は、図2の最後の文例で、入力から出力まで3秒程度である。

入力 > 今日の会議を知りたい。

出力 > イベント名：第2回部長会議
開催時間：3月28日 10時～12時
開催場所：第2会議室

イベント名：新製品開発会議
開催時間：3月28日 13時～14時
開催場所：第1会議室

;

入力 > 第2回部長会議は、いつ開かれますか。

出力 > 4月1日 13時から14時までに開かれます。

入力 > 来週の火曜日の午前に開かれる会議で、鈴木部長が出席するものがありますか。

出力 > 第1会議室で、10時から11時まで開かれる部長会議があります。

図2 試作インターフェイスの応答例

5. 意味解析の方法

5.1 意味解析における表現空間

意味解析とは、構文木の集合から、意味表現空間への変換を与えることであるととらえる。また、これらの空間の背後に意味ネットの空間があって、意味解析と密接に関係している。構文木の集合 Treeとしては、次のようなS式<tree>の集まりを考える。

<tree> ::= (<word> {<slot>}*)

<slot> ::= (<case> <tree>)

ここで、<word>とは、名詞、動詞、形容詞、形容動詞、副詞に対応するシンボルであり、<case>は表層格を指定するためのマーカである。このマーカは、助詞等を用いている。

意味ネットの空間 Semは、ノード間の関係として、二項関係のみをもつものとする。(ノードの集合をN、関係の集合をRとして、Sem=Sem(N, R)と表す。)このとき、名詞、動詞等の単語もすべて、Semのノードになっているととらえる。つまり、単語の集合をWとすると、

$W \subset N$

である。また、n が n' に対して、関係 r であることを

$n \xrightarrow{r} n'$

と表す。

意味表現空間 Repとして、次のようなS式<meaning>の集合を考える。Repは、すべての有意味な文章を解析して得られた意味表現形全体と考える。

<meaning> ::= (<primitive> {<attribute>}*) | (<concept>)

<attribute> ::= (<attribute-name> <meaning>)

<primitive>はターゲットシステムの処理やデータを表現するための基本的な概念を表

すシンボルであり、 $\langle \text{attribute} \rangle$ はその属性を表す。また、 $\langle \text{concept} \rangle$ は意味ネットのノードに対応していて、意味表現における属性値としてとらえる。また、 $\langle \text{primitive} \rangle$ はSemに埋め込まれているとする。つまり、 $\langle \text{primitive} \rangle$ の集合をPrmとすると、

$$\text{Prm} \subset N$$

である。

また、 $\langle \text{attribute-name} \rangle$ は $\langle \text{primitive} \rangle$ の属性名を表すものであり、各 $\langle \text{primitive} \rangle$ に対して、いくつかの $\langle \text{attribute-name} \rangle$ が対応している。今回ターゲットとしたスケジュールシステムの $\langle \text{primitive} \rangle$ と $\langle \text{attribute-name} \rangle$ の一部を表1に示す。

$\langle \text{primitive} \rangle$	$\langle \text{attribute-name} \rangle$	意味
search	s-object	検索
event	event-name	イベント名
	event-time	開催時間
	event-place	開催場所
	event-attendant	出席者
	event-purpose	議題
resource	resource-name	!
!	!	

表1 $\langle \text{primitive} \rangle$ の例

更に、ある $\langle \text{primitive} \rangle$ p に対応する $\langle \text{attribute-name} \rangle$ の集合を、 $\text{attr}(p)$ で表す。

ここで、

$$p, p' \in \text{Prm}, p \xrightarrow{\text{isa}} p'$$

としたとき、

$$\text{attr}(p) \supset \text{attr}(p')$$

であるとする。

$\langle \text{attribute} \rangle$ は、属性を表す。従って、意味表現における順序は任意である。そこで、Repに次の同値関係を導入しておく。

$m, m' \in \text{Rep}$; $m = (p \text{ attr}_1 \dots \text{attr}_k)$, $m' = (p' \text{ attr}_1' \dots \text{attr}_k')$ として、 m' が m に同値とは、次の二つが成り立つことである。

(i) $p = p'$

(ii) $k = h$ かつ $(\text{attr}_1 \dots \text{attr}_k)$ は、 $(\text{attr}_1' \dots \text{attr}_k')$ を並べかえたもの。

m が m' に同値であることを $m \sim m'$ で表す。

5. 2 意味解析処理を表す写像

意味解析は、TreeからRepへの写像 Φ で表現される。

$$\Phi: \text{Tree} \rightarrow \text{Rep}$$

konoNUSでは、ある単語に対して修飾が行われるということは、その単語が表す概念の属性値を具体的にセットすることであると考え、意味解析の基本処理とし採用した。例えば、「今日の第1会議室での会議」といった場合、「会議」という概念の「開催時間」という属性に「今日」がセットされ、「開催場所」には「第1会議室」がセットされる。この属性は、各単語ごとに違うので変換規則も単語ごとに記述する必要があり、 Φ を各単語ごとに記述する方法を採る。5. 1で述べた単語の集まりをWとし、 $\omega \in W$ に対して、

$$\text{Slot}(\omega) = \{(\text{slot}_1, \dots, \text{slot}_n); (\omega \text{ slot}_1 \dots \text{slot}_n) \in \text{Tree} \quad n=0,1,2,\dots\}$$

とする。この $\text{Slot}(\omega)$ は、 ω を修飾する言葉の集まりである。このとき、 $\text{Slot}(\omega)$ からRepへの写像 Φ_ω を次のように定義する。

$$\Phi_\omega(s) = \Phi((\omega \text{ slot}_1 \dots \text{slot}_n)) \quad s = (\text{slot}_1, \dots, \text{slot}_n) \in \text{Slot}(\omega)$$

この Φ_ω を $(\omega \text{ 上})$ 意味写像という。また、 Φ_ω は次のようにも表される。

$$\Phi_\omega(s) = (\Psi_\omega(s) X_{\omega^1}(s) \dots X_{\omega^m}(s))$$

ここで、 Ψ_ω は、 $\text{Slot}(\omega)$ からPrmへの写像である。

辞書には、この Φ_ω を記述すればよい。ただし、格文法の立場をとるので、 $(\text{slot}_1, \text{slot}_2, \dots, \text{slot}_n)$ を並べかえたものが $(\text{slot}_1', \text{slot}_2', \dots, \text{slot}_n')$ であるとき、

$$\Phi_\omega(\text{slot}_1, \text{slot}_2, \dots, \text{slot}_n) = \Phi_\omega(\text{slot}_1', \text{slot}_2', \dots, \text{slot}_n')$$

であるとする。一般に、このような性質を持つ $\text{Slot}(\omega)$ からRepへの写像を $(\omega \text{ 上})$ 表現写像と呼ぶことにする。

上のように、各語ごとに変換規則を記述するのは、面倒である。しかし、語の修飾が属

更に、一つの単語 ω 固有の表現写像 ϕ_ω も、同様に分解できる。例えば、「会議」という言葉に「<tm>の」（<tm>は時間を表す言葉）、「<pl>での」（<pl>は場所を表す言葉）といった修飾があった場合、これらは独立に処理されて、「会議」の別々の属性をセットする。従って、「<tm>の」を処理する変換を $T_{\text{会議}}$ 、「<pl>での」を処理する変換を $X_{\text{会議}}$ とし、「会議」に対して、このような修飾しかないとするれば、 $\phi_{\text{会議}} \sim T_{\text{会議}} \otimes X_{\text{会議}}$ と表記される。これに関係して更に、次の考え方を導入する。

ϕ_ω を ω 上の表現写像とする。

(i) ϕ_ω が可約とは、 ω 上の表現写像 χ_ω, τ_ω ($\neq \phi_\omega$) が存在して

$$\phi_\omega = \chi_\omega \otimes \tau_\omega$$

と表わされることである。 $\chi_\omega \otimes \tau_\omega$ を ϕ_ω の分解といい、 χ_ω, τ_ω を ϕ_ω の成分という。

(ii) ϕ_ω が可約でないとき、 ϕ_ω を既約という。

(iii) 全ての成分が既約である分解のことを既約分解という。

konoNUSでは、各単語毎の写像 ϕ を既約分解して、記述している。

また、上の説明では、単語の概念的意味は一つであるとした。しかし、実際には、修飾語により、複数の排反する意味を意味をもつ場合もある。ある意味には、表現写像 ϕ_ω が、別の意味には、 ϕ'_ω が対応していたとすると、全体の表現写像を

$$\phi_\omega \oplus \phi'_\omega$$

で表す。これは、 ϕ, ϕ' のどちらか一方の写像のみを適用することを意味する。

このオペレータ $\otimes, \oplus$ で、変換処理を構造化する。これらのオペレータには、次のような性質がある。

(i) 可換則 $\phi_\omega \otimes \phi'_\omega \sim \phi'_\omega \otimes \phi_\omega, \phi_\omega \oplus \phi'_\omega \sim \phi'_\omega \oplus \phi_\omega$

(ii) 結合則 $\phi_\omega \otimes (\phi'_\omega \otimes \phi''_\omega) \sim (\phi_\omega \otimes \phi'_\omega) \otimes \phi''_\omega$

$$\phi_\omega \oplus (\phi'_\omega \oplus \phi''_\omega) \sim (\phi_\omega \oplus \phi'_\omega) \oplus \phi''_\omega$$

(iii) 分配則 $\phi_\omega \otimes (\phi'_\omega \oplus \phi''_\omega) \sim (\phi_\omega \otimes \phi'_\omega) \oplus (\phi_\omega \otimes \phi''_\omega)$

これらの性質を用いると、表現写像がコンパクトに記述できる。例えば、ある言葉に対して、ある一つの修飾があるか、別の修飾があるかどうかで、まったく違う意味になるが、他の修飾に対する処理は同じ場合がある。つまり、

$$(\phi_\omega \otimes \phi^1_\omega \otimes \dots \otimes \phi^n_\omega) \oplus (\phi'_\omega \otimes \phi^1_\omega \otimes \dots \otimes \phi^n_\omega)$$

と表現される場合は、同値な表現

$$(\phi_\omega \oplus \phi'_\omega) \otimes \phi^1_\omega \otimes \dots \otimes \phi^n_\omega$$

を用いて、よりコンパクトな記述になる。

5. 4 意味解析言語 konoNUL

konoNUSでは、このような処理をオブジェクト指向型⁴⁾の意味解析言語konoNULを用いて記述している。konoNULは、次の特徴をもっている。

(1) 各オブジェクトを意味ネットのノードとしてとらえ、意味ネットの関係を記述する。

(2) 意味ネットのisa関係により、オブジェクトの親子関係を与える。

(3) パターンマッチングを拡張した「概念マッチング」(後述)によるメッセージセレクトがある。

(4) オブジェクト間では、メッセージのリストを送受信する。メッセージリストにおけるメッセージの順序は意味を持たない。

(5) 多重継承をサポートしている。

(6) 前節で述べた構造化オペレータ $\otimes, \oplus$ をサポートする記法がある。

意味解析においては、各単語をオブジェクトとみなす。そして、各オブジェクト ω ($\omega \in W$) に意味写像 ϕ_ω を記述する。構文木 ($\omega \text{ slot}_1 \dots \text{slot}_n$) の解析は、オブジェクト ω に $\text{slot}_1, \text{slot}_2, \dots, \text{slot}_n$ がメッセージとして送られる形で実現される。メッセージの解析は、次のようなルールが基本単位となっていて、ひとつのルールが前節で述べた既約な表現写像に対応している。

$$\text{IF } P_1 \ P_2 \ \dots \ P_n \ \text{THEN } Q$$

P_i ($i=1, 2, \dots, n$) はメッセージに対するマッチングパターンであり、 Q は結果の意味表現パターンである。例えば、「会議」に対する意味解析処理として、「今日の会議」というように時間を表す言葉（「時間」の下位概念となっている言葉）で修飾されたら、「会議」の「開催時間」の属性に対して、「今日」をセットするといった処理を記述する。このために、「時間を表す言葉に対するマッチング」といった概念的なマッチングが必要である。konoNULでは、次のような「概念マッチング」を行っている。

性をセットすることであると考えると、オブジェクト指向型手法を用いて、記述効率を著しく高めることができる。以下にこれを述べる。

5.3 構造化オペレーションとオブジェクト指向型手法

まず、説明のために次の定義をする。

$\omega \in W$ について、 ω 上の表現写像、 ϕ_ω, ϕ_ω が与えられたとする。このとき、 ϕ_ω が ϕ_ω に同値であるとは、次が成り立つことである。

$\forall s \in \text{Slot}(\omega) \quad \phi_\omega(s) \sim \phi_\omega(s)$ (4.1 で述べた Rep における同値)
 ϕ_ω と ϕ_ω が同値であることを

$$\phi_\omega \sim \phi_\omega$$

と書く。また、次の条件 (i) (ii) が満たされるとき、 ϕ_ω と ϕ_ω の積 $\phi_\omega \otimes \phi_\omega$ を以下のようにして定義する。

$$\phi_\omega(s) = (\alpha_\omega(s) \ \beta_\omega^1(s) \ \cdots \ \beta_\omega^n(s))$$

$$\phi_\omega(s) = (\delta_\omega(s) \ \gamma_\omega^1(s) \ \cdots \ \gamma_\omega^m(s))$$

と表したとき、

$$(i) \ \alpha_\omega(s) \stackrel{\text{isa}}{=} \delta_\omega(s), \text{ または } \delta_\omega(s) \stackrel{\text{isa}}{=} \alpha_\omega(s)$$

ただし、 $n, n' \in N$ について $n \stackrel{\text{isa}}{=} n'$ とは、

$\{ \exists n'' \in N, n \stackrel{\text{isa}}{\rightarrow} n'' \text{ かつ } n'' \stackrel{\text{isa}}{=} n' \}$ または $\{ n \stackrel{\text{isa}}{\rightarrow} n' \}$ であること。

$$(ii) \ \varepsilon_\omega(s) = \begin{cases} \alpha_\omega(s) & (\alpha_\omega(s) \stackrel{\text{isa}}{=} \delta_\omega(s) \text{ のとき}) \\ \delta_\omega(s) & (\delta_\omega(s) \stackrel{\text{isa}}{=} \alpha_\omega(s) \text{ のとき}) \end{cases}$$

のとき、

$$\forall s \in \text{slot}(\omega) \quad (\varepsilon_\omega(s) \ \beta_\omega^1(s) \ \cdots \ \beta_\omega^n(s) \ \gamma_\omega^1(s) \ \cdots \ \gamma_\omega^m(s)) \in \text{Rep}$$

このとき、

$$\phi_\omega \otimes \phi_\omega(s) = (\varepsilon_\omega(s) \ \beta_\omega^1(s) \ \cdots \ \beta_\omega^n(s) \ \gamma_\omega^1(s) \ \cdots \ \gamma_\omega^m(s))$$

により $\phi_\omega \otimes \phi_\omega$ を定義する。

また、話を簡単にするため、次のことを仮定する。

$\omega \in W$ は修飾語によらず、ひとつだけの意味をもつ。つまり、

$s \in \text{Slot}(\omega)$ とすると、

$$\Phi_\omega(s) = (C_\omega \ X_\omega^1(s) \ X_\omega^2(s) \ \cdots \ X_\omega^n(s)) \quad (C_\omega \text{ は } s \text{ には依らない})$$

である。($X_\omega^i(s)$ が空列の場合も含む。)

さて、 $\omega, \omega' \in W$ について、 $\omega \stackrel{\text{isa}}{\rightarrow} \omega'$ のとき、 ω は ω' の下位概念であるから、 ω は ω' より多くの属性をもっている。従って、 ω に対する修飾も ω' より多様であると考えられるから、

$$\omega \stackrel{\text{isa}}{\rightarrow} \omega' \text{ ならば、 } \text{Slot}(\omega) \supset \text{Slot}(\omega'),$$

が成り立つとする。また、 C_ω の属性は、 $C_{\omega'}$ の属性に C_ω 固有の属性を加えたものであるから

$$\text{Attr}(C_\omega) \supset \text{Attr}(C_{\omega'})$$

である。従って、 C_ω 固有の属性をセットする表現関数を ϕ_ω とし、

$$\phi_\omega(s) = (C_\omega \ X_\omega^1(s) \ \cdots \ X_\omega^n(s))$$

$$\Phi_{\omega'}(s) = (C_{\omega'} \ Y_{\omega'}^1(s) \ \cdots \ Y_{\omega'}^m(s))$$

と表せば、

$$\Phi_\omega(s) = (C_\omega \ Y_{\omega'}^1(s) \ \cdots \ Y_{\omega'}^m(s) \ X_\omega^1(s) \ \cdots \ X_\omega^n(s))$$

と表される。さらに、 ω は ω' の下位概念であるから、 $\langle \text{primitive} \rangle$ の空間 Prm においても、この関係は保たれて、

$$C_\omega \stackrel{\text{isa}}{=} C_{\omega'}$$

が成り立つとする。以上のことにより、 Φ_ω は次のように分解される。

$$\Phi_\omega \sim \phi_\omega \otimes \Phi_{\omega'}$$

従って、 ω の意味解析処理としては ϕ_ω のみを記述しておけば、 ω' 上の意味写像 $\Phi_{\omega'}$ と合わせて、 Φ_ω を求めることができる。

konoNUS では、 W を意味ネット Sem の中に埋め込み、更に Sem の各ノードに、それ固有の表現写像をセットしている。

5.5 概念マッチング

$\text{Sem}(N, R)$ を意味ネットとする。ただし、 R は、 N 上の二項関係の集合である。 $P(N)$ で N の部分集合全体を表すとき、 R は $P(N)$ から $P(N)$ への写像の集合と見なせる。それは、以下のようにすればよい。まず、

$r \in R, n \in N$ に対して

$$\hat{r}(n) = \{n' \in N \mid n \xrightarrow{r} n'\}$$

とする。 $\hat{r}$ は N から $P(N)$ への写像となる。更に、 $A \in P(N)$ として

$$\hat{r}(A) = \bigcup_{n \in A} \hat{r}(n)$$

とすると、 $\hat{r}$ は $P(N)$ から $P(N)$ への写像となる。 r から $\hat{r}$ への対応を α で表すと、

$$\alpha : R \rightarrow M(N) \quad (\text{ただし、} M(N) = \{f : P(N) \rightarrow P(N)\})$$

容易に証明できるように

$$\alpha(r) = \alpha(r'), (r, r' \in R) \Rightarrow r = r'$$

であるので、 α により R を $M(N)$ の部分集合と見なすことができる。

ここで、概念マッチングを次のように定義する。

$a, b \in N, r \in R$

a が b に r でマッチするとは、ある n ($n = 0, 1, 2, \dots$)があつて

$$r \circ \text{isa} \circ \dots \circ \text{isa} (\{a\}) \supset \{b\}$$

$\underbrace{\hspace{1.5cm}}_{n \text{ 個}}$

が成り立つことであり、 $a \stackrel{r}{\models} b$ と書く

特に、 a が b に isa でマッチするときには、単に a が b に概念的にマッチするという。

5.6 konoNULでの記述

konoNULでは、メッセージに対するマッチングに概念マッチングを用いている。

具体的には、

$$a \stackrel{r_1}{\models} b_1 \text{ かつ } a \stackrel{r_2}{\models} b_2 \text{ かつ } \dots \text{ かつ } a \stackrel{r_n}{\models} b_n \text{ を}$$

$$a : r_1/b_1 \quad r_2/b_2 \quad \dots \quad r_n/b_n \text{ と書く。}$$

例えば、「時間を表す言葉」は、

$*t : \text{isa/time}$ (または、 isa/ を省略して $*t : \text{time}$) ($*t$ はパターン変数を表す)と書く。

「会議」を「時間を表す言葉」が修飾している場合の解析ルールは

IF (no ($*t : \text{time} . *rest$)) THEN (event-time $\langle \$send(*t . *rest) \rangle$)

となる。 $\langle \$send(*t . *rest) \rangle$ は、 $*t$ に $*rest$ をメッセージとして送信して返された結果を表す。

更に、先に述べた構造化オペレータ $\otimes, \oplus$ の機能をサポートする記法として、 $\{af \dots\}, \{ff \dots\}$ ($af = \text{all fitted}, ff = \text{first fitted}$)を用意している。

5.7 意味解析における二つの層

言葉の意味を考えると、構文上の情報から反映される部分と、単語固有の意味から反映される部分がある。konoNULでは、意味解析オブジェクトを構文層と意味層の二つの層に分けた。助詞「と」などを用いた場合のように、意味に反映するような構文に対する処理は構文層に記述する。

konoNULの解析では、単語 ω に対する修飾を解析する場合、まず、構文層の ω にメッセージが送られて、その後に、意味層の ω にメッセージが転送される。

構文層での処理は、構文が意味に与える影響を解析するだけであるので、ターゲットシステムに依存しない。

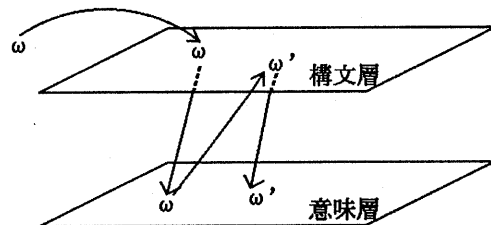


図3 構文層と意味層

5.8 意味解析処理における係り受けの修正

konoNULのドライバでは、解析できなかったメッセージをもとのオブジェクトに返すようにしている。つまり、オブジェクトAからオブジェクトBにメッセージ $\{m_1, m_2, \dots, m_n\}$ が送られたとする。Bでは、 $\{m_1, \dots, m_i\}$ ($1 \leq i \leq n$)がルールにマッチして、意味表現Pが得られたが、残りの $\{m_{i+1}, m_{i+2}, \dots, m_n\}$ はマッチするルールがなかったとする。このとき、

B から A に、 $\{m_{i+1}, m_{i+2}, \dots, m_n\}$ が返され、B で再処理される。

例えば、「部長の来週の予定」という文章を考えてみる。「部長の」は、「来週」ではなく、「予定」に係っている。意味的に考えて、「部長の」は「来週」には係り得ない。これは、「来週」には「部長の」による修飾によってセットされるべき属性がないからである。konoNUS の文法では、「部長の」は「来週」に係るとして、構文木が生成される。

しかし、konoNUL による意味解析の過程において次のようにして、正しい意味表現が生成される。

- ①オブジェクト「予定」には (no (raisyyu (no (butyou)))) がメッセージとして送られる。
- ②予定に書かれたルールにより、「来週」に (no (butyou)) が送られる。「来週」には (no (butyou)) を解析するルールは書かれていない。
- ③「来週」からは、「来週」に対応する意味表現形 P_1 と (no (butyou)) が「予定」に返される。
- ④「予定」では (no (butyou)) をもう一度解析しなおす。「予定」では (no (butyou)) にマッチするルールがあって「部長」に空メッセージが送られ、「部長」では対応する意味表現形 P_2 が与えられる。(図4 参照)

最終的には、初めから、予定に (no (butyou)), (no (raisyyu)) が送られたのと同じ結果 P_1, P_2 が得られる。

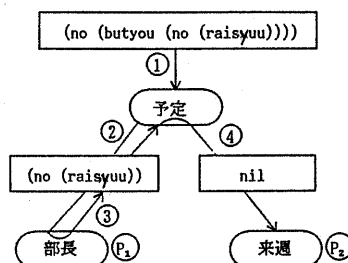


図4 係り受けの修正

6. おわりに

スケジュールシステムに対するインターフェイスを試作することにより、konoNUL の意味解析言語としての有効性を確認できた。しかし、記述性等、課題として残されているものも多い。今後は、konoNUL の改良と共に、開発環境の整備に取り組んでいく予定である。

【参考文献】

- [1] 角田 透, 中村 輝雄, CP/M68k用Lisp (konoLisp) のLisp 言語による開発
記号処理研究会資料 31-12 1985
- [2] 長尾 真, 言語工学 昭晃堂 1983
- [3] Griffith.R.L, Three Principles of Representation for Semantic Networks
ACM TODS Vol.7 No.3 1982
- [4] Goldberg.A, Robson.D, Smalltalk-80 The Language and its Implementation
Addison Wedsley 1983
- [5] 大沢 一郎, 米澤 明憲, オブジェクト指向方式による対話理解システム
自然言語研究会資料 44-7 1984