自然言語処理 31－5
（1982. 5. 21）

意味表現用言語SRLの機械翻訳への応用
―融合方式の提案―

田中穂積，元吉文男（以上 電子技術総合研究所），安川秀樹（松下電器）

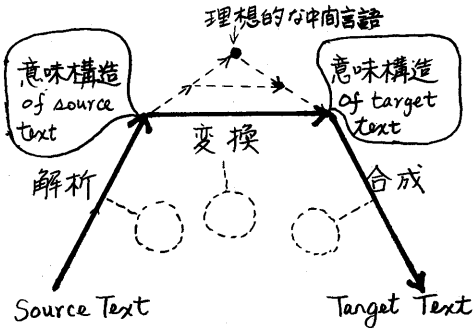
1. はじめに

筆者等は，日本語の意味構造を抽出するプログラムEXPLUSを作成した[田中79]。EXPLUSでは，フレーム形式の意味表現用言語SRLを提案し，その有効性を確認した。以下では，SRLが機械翻訳に応用可能なことを示す。機械翻訳で重要な問題の一つに，訳語の選択をどの様に行うかがある。この問題に対する一つの解が以下で示されるであろう。この問題を避けて，一つの単語に一つの固定した訳語しか対応させえない機械翻訳の方式は，原理的に統語的な処理に基づく機械翻訳の方式であるといって良いだろう。訳語の選択には，意味解析が大きな役割を果たすからである。

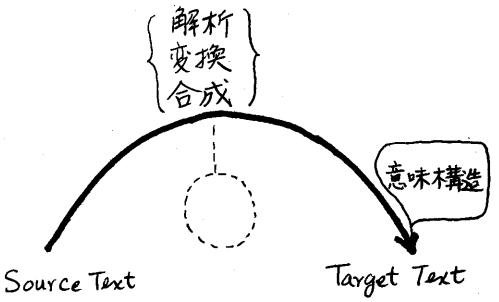
2章以下で説明する筆者等の機械翻訳の方式は，次の特徴を有す。

(i) 意味解析を行い，意味構造を抽出する過程で訳語を選択してしまう。
(ii) 意味解析の過程で翻訳文の語順を決定してしまう。
(iii) 意味解析（意味構造の抽出）が終了するとともに，翻訳文が合成される。

(iii)は，(i)と(ii)からの自然な帰結であるが，これは，従来のトランスファー方式（図1a）と著しく異なる。図1bに示す様に，意味解析過程と訳語選択過程，翻訳文合成過程とが強く融合した方式である。その意味で筆者等の方式を，以下では融合方式とよぶことにしたい。これは，意味解析を，構文解析木に沿って行うので，木構造の直接的な変換を行う必要がない。SRLの枠組のなかに組み込まれた人工知能の手法を利用した方式である。また，



(a) トランスファ方式



(b) 融合方式

図1. トランスファ方式と融合方式の概念図

意味解析を中核に据えた翻訳文の合成方式である。筆者等が作成した機械翻訳実験システムは，拡張Lingol上に作成されている[田中他77]。[田中82]で説明した様に，漢字Prologが作成されれば，それでこの実験システム全体を書き直すことを検討している。

2. 意味表現用言語SRLの概要

融合方式の説明に入る前に，意味表現用言語SRLの概略を説明する．SRLは，日本語の意味構造を抽出するために考えられた手法である。[田中 79]では，これは英語等の他の言語の意味

(a) <unit記述> ::=
(<unit-desig> unit
<part-of>
<self>
<sem-feature>
<slot>
.....
<slot>)

(b) <unit-desig> ::= (<unit-name> <訳語>) |
((<unit-name> <訳語>) <index>)

(c) <slot> ::= <satisfied-slot> | <unsatisfied-slot>
<satisfied-slot> ::= (<slot-name> = <value>)
<unsatisfied-slot> ::=
(<slot-name>
(<constraint> • <prep>) • <action>)

図2. 翻訳用辞書のSRLによる<unit記述>のシンタクス

構造の表現にもそのまま適用可能なことを指摘しておいた。

SRLで意味表現の基本は<unit記述>である。詳細は[田中 79]を参照していただきたい。ここでは以下の英日翻訳システムで必要となる事柄のみを説明する。

図2の<unit記述>中の<part-of>, <self>は、それぞれ全体/部分、上位/下位の関係を記述し、このユニットが他のユニットとどの様に意味的に関連するかを記述するものである。<sem-feature>には、そのユニットの意味素性や統語的素性を書く。

図2に示す様に、英日翻訳用辞書のSRL記述中の<unit-desig>中には、<訳語>が付加されている。これは意味解析が進むにつれて適切な訳語に変換されるもので、デフォールトな訳語であることに注意されたい(後述)。

SRLによる<unit記述>中で、融合方式による機械翻訳で最も重要な役割を果たすものは、図2(c)の<unsatisfied-slot>である。<slot-name>には、subj obj, prepと書かれ、このスロットを満たしうるユニット(以下ではFILLERとよぶ)の統語的制約条件を記す。

<constraint>は、FILLERの意味的制約条件を書く。SRLではここに、必要に応じて非常に詳細な、and, or, notを組み合せた意味制約条件を書くことができる[田中 79]のが特徴である。必要ならここに to-test method として evaluable predicate を Lisp のS式として書くこともできる。

<prep>は[田中 79]では、<J-case>(Japanese-case)とよばれ、FILLERに後続すべき日本語の助詞(後置詞)を指定した。図2は英日翻訳用の辞書項目の記述だから、ここは<prep>(Preposition)として、FILLERに前置されているべき前置詞を書く。この統語的制約条件の検査が不要の場合には、<J-case>と同様、ここにアンダーライン"_"を記せばよい。

<action>は、<slot-name>, <constraint>, <prep>の適当な組み合せによる制約条件検査に合格したFILLERが、このスロットを満たした時点で起動するアクションを記述する。EXPLUS[田中 79]では、<action>の起動により、意味構造抽出過程の制御や、より深い意味構造の抽出が行われた。融合方式ではその他に、<action>の起動により適切な訳語選択を行う。なお<action>は省略も可能であり、複数個並記してもよい。<unsatisfied-slot>は一つのプロダクション規則と見なしうるので[田中 79]、<unit記述>は、プロダクション規則の集合に、特殊なスロット(<part-of>, <self>, <sem-feature>)が付加されたものと見なすことができる。

図3に、筆者等が用いている英日機械翻訳用の辞書項目の記述例を示す。これは、拡張Lingolの形式で記述されており、その<sem>部[田中

79]が，SRLによる<unit 記述>になっている。したがって図3の4行目以下がSRLによる“open”の<unit 記述>である。　図3の10行目以降に<unsatisfied_slot>が列記されている。

たとえば第19行目の<unsatisfied_slot>では，

<slot_name> : obj;
<constraint> : (OR (a DOOR) (a WINDOW));
<prep> : —;
<action> : (@ZYOSHITSUKE FILLER 'DE),
(@YAKUGO ORIGIN '(開け・RU));

である。

以上の記述から，この<unsatisfied_slot>に対するFILLERの意味的制約条件がドアもしくは窓((OR (a DOOR) (a WINDOW)))であり，その統語的制約条件が目的格(obj)で前置をとらない(—)ことを記述することができる。<action>は，@ZYOSHITSUKEと@YAKUGOという名称の二つのLisp関数で，FILLERによってこのスロットが満たされた時点で順に実行される(次章参照)。

図3の10行目と19行目のスロットの組み合せから“Human opens a door or a window”に対する訳文が，“人がドアもしくは窓を**開ける**”，であることが読み取れる。これは一つの翻訳用例文が与えられているとみなせる。ORIGINは，この<unsatisfied_slot>を含むユニットを指す。10行目と19行目と35行目のスロットの組み合せから，“Human opens a door or window with an instrument or a hand”に対する訳文が，“人が道具もしくは腕でドアもしくは窓を**開ける**”であることが読み取れる。11行目のスロット単独からは“The door opens”に対する訳文が“ドアが**開く**”であることが読み取れるが，これはデフォールト訳語の説明から明らかだろう。

```
(V
  OPEN
  (NIL  0)
  '((OPEN  開  .  KU)
      unit
      (part-of)
      (self
          (a  DEFAULT-CASE))
      (sf  +action)
      (subj  ((a  HUMAN)  -))
      (subj
          ((OR
                (a  DOOR)
                (a  WINDOW))
            -))
      (subj
          ((a  INSTRUMENT)
            -))
      (obj
          ((OR
                (a  DOOR)
                (a  WINDOW))
            -)
          (@ZYOSHITSUKE
              FILLER
              'WO)
          (@YAKUGO
              ORIGIN
              '(開け  .  RU)))
      (obj
          (T  -)
          (@ZYOSHITSUKE
              FILLER
              'WO))
      (prep
          ((OR
                (a  KEY)
                (a  ARM))
            -WITH)
          (@ZYOSHITSUKE
              FILLER
              'DE)
          =instrument)
      (prep  (T  -))))
```

図3　openの辞書記述

3. 融合方式による機械翻訳*

機械翻訳に，意味解析が必要なことは明らかである。筆者等は，日本語の意味解析を行うプログラムEXPLUSを開発してきた。このプログラムをそのまま利用して，文章の意味を解析しながら訳語の選択を行い，合わせて目標言語の語順に合った文を合成する方法を提案し，これを融合方式とよぶことにした。この方式による機械翻訳では，第1章で述べた様に，意味解析の終了と共に機械翻訳結果を得るもので，意味解析と明確に分離したフェイズとしてトランスファー過程を有する方式と大いに異なる。

そこで，融合方式の基本となるユニット間会話の考え方に基づく意味解析と，その過程で語順の変更をどの様に行うかを説明する。

* 以下では英日機械翻訳を例にとり説明する。推論機構研究室では，融合方式による日英機械翻訳の実験を進めており有望な結果を得ている。これについては別稿で報告の予定である。

3.1 ユニット間会話と訳語の選択

ある語の訳語を決めるためには，その語がどの様なコンテキストにおかれているかを知る必要がある。そのために筆者等がEXPLUSで提案したユニット間会話の考え方を利用する。ユニット間会話は，構文解析木の構造に沿って行う。

図4に，"He opens the door with a key." という文を構文解析した結果を示す。

ユニット間会話の基本は，2つのユニットのうちの片方が，他のユニット中のいずれかのスロットを満たしうるかどうか調べるものである。ユニット間会話では，前者をFILLER，後者をORIGINとよんでいる。FILLERがORIGIN中のいずれかのスロットを満たすためには，2章で説明した<unsatisfied_slot>中の統語的制約条件（<slot_name>と<prep>）と意味的制約条件（<constraint>）の適当な組み合せを，FILLERが満たさねばならない。

FILLERがORIGIN中のいずれかのスロットを満たすことは，FILLERの指示するコンテキストがORIGINに対して付加されることを，またその逆にORIGINの指示するコンテキストが，FILLERに対して付加されることを知る。以下に述べる様に，この機会はコンテキストに依存した訳語選択の機会を与える。

たとえば図4の点線で囲まれた部分は，VP → V NP という文法規則を適用して得られた部分木である。この部分木の節点Vと節点NPとには，それぞれ"open"と"door"の辞書項目から得られたユニット記述が送られてきている。この点線のなかの節点VPでは，下位の節点Vに送られてきているユニットをORIGINとし，下位の節点NPに送られてきているユニットをFILLERとしてユニット間会話を行う。

これが具体的にどの様なプログラムとして実現されているかを図5に示す。

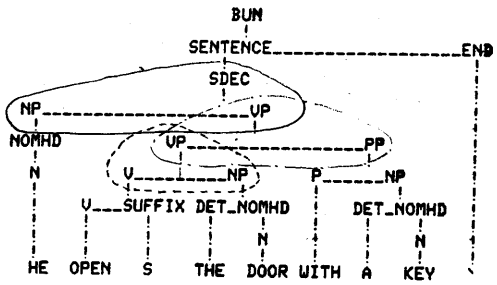


図4　構文解析例

```
(VP ((V ...)                                   } VP → V・NP
    (NP ...))
0.
((LAMBDA (NP V @TRANS)                         (a) 指定の語順に従うユニットの並びかえ
   (SETQ @TRANS                                (b) ユニット間会話
         (@TRANSLATION @TRANS))                (c) 節点VPへ送られる情報
   (@JOIN0 'OBJ                                ラムダ式の変数への値を束縛
           (CAAR NP)                           <eom> 部
           'obj
           (CAAR V))
   (@TR_FORM1 V @TRANS))
 (DO 'NP)                                      ---- 節点NPのユニット
 (DO (@COPY_U (DO 'V)))                        ---- 〃 V 〃
 '(NP V))                                      ---- 語順指定
```

図5　文法規則の例

図5の関数@JOINOがユニット間会話を行う関数である。@JOINOでは，NPに送られてきているユニットが，Vに送られてきているユニットのobjスロットを満たすかどうか調べる。図4の場合，doorのユニットがNPに送られてきており，openのユニットがVに送られてきているので，doorのユニットをFILLERとして，図3の19行目以下のobjスロットが順に調べられる。たまたま，19行目の最初のobjスロットの<constraint>は(OR (a DOOR) (a WINDOW))であるから，このFILLERによって満たされ，2つの<action>，(@ZYOSHITSUKE FILLER 'WO)

と（@YAKUGO ORIGIN `(開け・RU))が実行される。

前者はFILLER(doorユニット）に助詞“を”を付加する関数である。後者はORIGIN（openユニット）の訳語を，デフォールト訳語“開く”（図3参照）から“開ける”に変更する関数である。

この様にして，ユニット間会話により，適切な訳語の選択がなされるだけでなく，適切な助詞の選択付加もなされる。

図5の(c)の@TR-FORM1では，上位の節点VPに，節点Vの情報のみを送ることを指示する。

以上の説明から，どのユニットとどのユニットとを，どの様に会話させ，どの様な会話結果を上位の節点に送るかは，図4に示す構文解析木を構成する各文法規則の<sem>部にどの様なプログラムが書かれているかによって決まることが分かる。

一点鎖線で囲まれた文法規則に付加された<sem>部では，下位の節点Vに送られてきたopenユニットをORIGINに，下位の節点PPに送られてきた前置詞with付きのkeyユニットをFILLERにしたユニット間会話を行う。その結果，keyユニットは35行目からはじまるスロットを満たし，<action>として関数（@ZYOSHITSUKE FILLER `DE）を実行する。これによりkeyユニット（FILLER）には，助詞“で”が付加される。このユニット間会話では，<prep>が利用される。

一点鎖線で囲まれた部分木の上位の節点VPには，再びopenユニットが送られ，これは実線で囲まれた節点NPのheユニットをFILLERとしたユニット間会話をひき起こす。heユニットは“human”だから，図3の10行目のsubjスロットを満たす。この場合の部分木はSDEC → NP・VPという文法規則から構成されているので，NPはVPの主格（subj）であることが知れる。そこでSDEC → NP・VPに付加された<sem>部のプログラムでは，<slot_name>としてsubjを指定したユニット間会話を行う。（説明が前後するが，図5の文法規則では，NPはVの目的格（obj）であることが分かるので，(c)のユニット間会話で，objを指定しているのである。同様に，先のkeyユニットとopenユニットとの会話は，文法規則VP → VP・PPに付加された<sem>部のプログラムで行うので，<slot_name>としてprepを指定したユニット間会話を行う）。

以上の様に，構文解析木に沿って，ボトムアップにユニットの転送とユニット間会話が繰り返されて，図6（下図）の

```
**********************
(((HE 彼 . HA) 4)
    unit
    (part-of)
    (self (a HUMAN))
    (sf))

**************************
**************************
(((KEY 鍵 . DE) 3)
    unit
    (part-of)
    (self (a INSTRUMENT))
    (sf -natural))

**************************
**************************
(((DOOR そのドア . WO) 1)
    unit
    (part-of)
    (self (a OBJECT))
    (sf -natural))

**************************
**************************
(((OPEN 開け . RU) 2)
    unit
    (part-of)
    (self (a DEFAULT-CASE))
    (sf +action)
    (subj = ((HE 彼 . HA) 4))
    (obj
        =
        ((DOOR
                そのドア
                . WO)
            1))
    (instrument
        =
        ((KEY 鍵 . DE) 3)))

**************************
```

図6 意味解析結果（意味構造）
（入力文 “He opens the door”）.

意味解析結果を得る。

図6意味構造を上から下にたどり，各ユニットの<訳語>を並べると，日本語の訳文になっている。これが，意味構造の抽出と同時に，翻訳文が合成されることの意味である。これについて次節で説明する。

3.2 語順の変更

これまでに説明したユニット間会話を利用した意味解析の手順をまとめると次の様になる。

(i) 入力文を構文解析し構文解析木を得る。
(ii) 構文解析木を構成する各文法規則に付加された<sem>部のプログラムを起動し，構文解析木の各節点で，下位から送られてきたユニット相互間で，プログラムの指定するユニット間会話を行い，会話の結果を上位に返す。
(iii) (ii)を繰り返し，構文解析木の最上位節点に文の意味解析結果を得る。

以上の一連の手順を実行する機会は文法規則に付加された<sem>部のプログラムによって与えられる。これは実は翻訳文(目標言語)の語順を決定する良い機会でもある。(ただし，意味解析の基本単位がユニットであるから，語順ではなく，ユニット順を決定すると言い換えるべきかもしれない。)

3.1では，ユニット間会話の結果として，上位の節点で行われるユニット間会話で必要な情報のみが選択的に上位に送られることを説明した。図4の点線で囲まれた部分木に対しては，上位の節点VPには，節点Vのopenユニットのみが送られ，次のユニット間会話で使われることになっていた。節点NPのdoorユニットは，openユニットのFILLERとして吸収され，見かけ上意味解析の場面から消える。

この様に，意味解析に必要なユニットが選択されて上位に送られるだけでなく，実は目標言語の文法に適った語順のユニットの集合も同時に上位節点に送られる。後者を語順変更用スタックとよぶことにする。上図は，図4の点線で囲まれた部分木の文法規則(図5)の<sem>部で行なうユニット間会話の様子を示している。

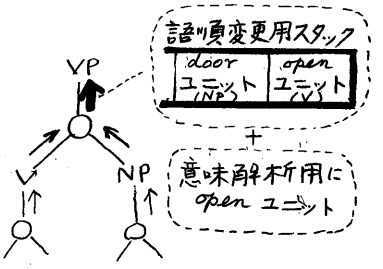


図5の文法規則は，NPがVの目的格であることを示している。日本語では目的格の後に動詞を後置した語順である。そこで，図5の最後で指定した語順'(NP V)に従って，ユニットの並べ変えを行う。これが図5(a)の部分のプログラムである。変数@TRANSが，上図の語順変更用スタックであり，結果は上図に示した通りである。

節点VPには，図5(c)の関数により，意味解析用のVユニット(openユニット)と，日本語の文法に適った語順をもつ語順変更用スタックとがリストされたものが送られる。

これはちょうど，図7(a)の部分木が，図7(b)の部分木に変形されたことと等価である。

以上の様にして，語順変更用スタックへの(ユニットの)プッシュの操作順を適当に制御することにより，語順の変更を容易に行える。語順の変更は，

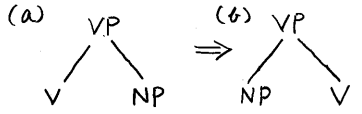


図7 語順変更用スタック(上図)の内容と等価な木の変形

英語の文法規則を参照して行なわねばならない。図5に示した様に，意味解析は，構文解析木の各部分木に対応する文法規則に付加された(<sem>部の)プログラムで行っているので，これはその文法規則を参照しつつ，目標言語である日本語の文法に適った語順を作り出す機会がこの時自動的に与えられることになるのである。従って，この方法は素直で自然な翻訳文合成法であると言える。

図5に従って考えると，一点鎖線で囲まれた部分木では，下位のVP節点に送られてきている語順変更用スタックに対して，*key* ユニットがプッシュされる。上位のVP節点の語順変更用スタックは

key ユニット	*door* ユニット	*open* ユニット

の様になる。更に図5の実線で囲まれた部分木では，このスタックに，*he* ユニットをプッシュする（SDEC → NP・VP から，NP（*he* ユニット）はVPの主格であることが分かるから，日本語も英語同様NP・VPの順に語を並べる）。結果は，図6に示したものになる。

3-3 *have a dance, have a drink*

動詞 *take* は，目的語に応じて様々な日本語へ訳し分けなければならない。これは3・1で説明した方法でほとんど解決できる。("*take*"訳し分け実験結果は[田中 82]を参照のこと）。しかし，いずれも目的語が最後の翻訳結果にあらわれていた。ところが *have a dance, have a swim, have a drink* 等は，単独の動詞 *dance, swim, drink* として訳すべきである。（前二者は目的語をそのまま残して，"踊りをする"，"水泳をする"の様に訳すこともできよう。しかしこれは *have a drink* には通じない）。

これは次のスロットを *have* の<*unit* 記述>にもたせることで容易に実現できる。

```
(V have ...
   '((HAVE 持・TTEIRU) unit
      .....
      (obj (+action -)
         (SETQ @TRANS
               (@REMO2 (LIST FILLER)
                       @TRANS))
         (@YAKUGO ORIGIN ....))
      ....))
```

(a) FILLER(目的語)を語順変更用スタック@TRANSから除去

(b) 自身の訳語をFILLERのactionスロットから取り出したものに変える。

ここでの<*action*>では，(a)目的語を翻訳結果から除去し，(b) *have* の<訳語>を，目的語の *action* スロットに書かれている<*value*>に変えることを指示している。

これにより"*I have a dance*"に対して

翻訳結果

私は 踊る。

を得ることができる。

4. おわりに

筆者等は，人間の介入をできるだけ減らした機械翻訳システムの実現を目指している。現在の実験システムでは"*I take my child to the bus stop with her*"という文からは4個の構文解析木を得る。その各々に意味解析を施すと，1つから

翻訳結果

私は 彼女と そのバス停のところへ 私の子供を 連れて行く。

を得て，他の3つからは

解釈不能 というメッセージが出力される。

3章までに説明した融合方式が，埋め込み文を含む英語の翻訳にも適用可能かどうかを知るために，[Robinson 82]の *introduction* の最初の文"DIAGRAM *is a grammar used in an artificial intelligence system for interpreting dialogue*"の文を借用して，"*Extended*

翻訳結果

拡張ＬＩＮＧＯＬは 対話を 解釈するために 人工知能システムで 使われる パーザー である。

翻訳結果

拡張ＬＩＮＧＯＬは 対話を 解釈するための 人工知能システムで 使われる パーザー である。

LINGOL is a parser used in an artificial intelligence system for interpreting dialogue" に対して得た翻訳結果を上に示す。この場合にはただ2つの構文解析結果を得て，その各々から2つの異なる日本語文が出力される。

ここでこれまで本文中で陽に指摘しなかったが重要と思われる事柄をまとめてみよう。1章の(i),(ii),(iii)で述べたことの他に本稿で提案した融合方式は：

(iv) 木の陽な変形を行わない。

(v) <constraint>の記述として意味素性もしくは意味マーカの粗いレベルの記述では不十分であり，場合によっては，self スロットを利用した上位/下位関係の利用と，and, or, not を組み合せた柔軟で精密な記述が必要である。SRLによれば，それが可能である。

(vi) たとえば，(日本語の)異なる訳語毎に take 等のユニット記述を行い，同音異義語を増やすことは，構文解析，意味解析を複雑にするので好ましくなく，本稿で提案した単一のユニット記述中の<action>による訳語選択を行う方法が優れていると思われる（詳しくは[田中 82]参照）。

(vii) 深層格パターンを全く利用せず，むしろ，subj, obj, prep 等の文法機能を翻訳の一部に利用している。

機械翻訳に関しては未解決の問題が山積している。それらについては，[長尾 82a]の考察を参照されたい。そこであげられていた A man eats vegitables と Acid eats metal の両文における eat の訳し分けは，3章で説明した方式で容易に可能であろう。Cheap wine flooded the market も正しく訳すことができるだろう。

2章では，SRLによる辞書項目記述により，幾つかの翻訳用例文が与えられていると見なしうる，という指摘がなされていた（"take"の辞書項目記述に関しては[田中 82]参照のこと）。これは(v)と合わせて，[長尾 82a]が4.3節で指摘した問題と関係している。

極めて限られた世界を対象にしているとはいえ，言語によらない普遍的な中間表現を介した翻訳も試みられている[Carbonell 81]。この様な方式と融合方式の利害得失もいずれ論じてみたい。Prologと機械翻訳の問題の一部は次回報告する。

最後に意訳の問題をSRL記法中のinferスロット[田中 79]を用いて部分的に解決する手法について，今少し熟慮して機会を見て報告してみたいと考えている。

謝辞 本システムは，推論機構研究室で開発された多くのプログラムを使用している。suffix処理用パッケージを書かれた井佐原均技官，ローマ字かな変換パッケージを書かれた横山晶一技官に感謝する。研究の機会を与えられた渕一博パターン情報部長に感謝する。

文献

[田中(他)77] "自然言語処理のためのプログラミングシステム―拡張LINGOLについて"，信学論誌，J60-D，12，1977，1061-1068.

[田中 79] "計算機による自然言語の意味処理に関する研究"，電総研研究報告797，1979.

[高松(他)81] "動詞パターンと格構造に基づく英日機械翻訳"，信学論誌D，9，1981，815-822.

[沢井(他)81] "知識表現FKR-0とその機械番翻訳への応用"，情報処理学会人工知能と対話技法 19-3，1981.

[McArthur 81] "Longman Lexicon of Contemporary English"，Longman，1981.

[Carbonell 81] "Steps Toward Knowledge-Based MT"，IEEE Trans. of PAMI，4，July，1981，376-392.

[長尾(他)82a] "機械翻訳に対するロングマン辞書データベースの応用"，情報処理学会NL研究会 29，1982.

[長尾 82b] "機械翻訳"，信学会誌 4，1982，386-392.

[田中 82] "自然言語処理技術研究への新たな展開に向けて"，情報処理学会人工知能と対話技法 25-3，1982.

[Robinson 82] "DIAGRAM: A Grammar for Dialogues"，Comm. of ACM，25，1，1982，27-47.

[新田(他)81] "英和機械翻訳のための構文解析技法" 情報処理学会NL研究会 28-4，1981