自然言語処理 43－3
（1984. 5. 24）

基本動詞「MAKE」を含む文の日本語への訳し分け

田中穂積、糸井理人 （東工大・工）

はじめに

電子技術総合研究所で開発された融合方式による英日機械翻訳システム[1]は、解析、変換、生成の3過程を融合した翻訳を行う。意味解析を行う過程で単語の訳し分けを行ない、意味解析が終了し、意味構造が抽出されるとともに翻訳結果が得られるという特徴がある。「TAKE」を含む英文についてはすでに20通り程度の訳し分けを行うことができることが示されていた。本研究ではこの特徴を生かし、「MAKE」について20数通りの訳し分けができるようにシステムを拡張した。「MAKE」は、目的語と同時に不定詞、名詞、形容詞の補語を伴い使役の意味を持つ。特に名詞の補語を伴うときは二重目的語とまぎらわしい、という点が「TAKE」と違い難かしい点である。また、動詞の訳語決定のために新しい機構を考案した。これにより動詞の訳語をその主語、目的語、前置詞句などの3個以上の語句を参照して定めることができるようになるとともに、より見通しの良い辞書記述が可能になった。

本稿では、まず辞書記述方式について説明し、次に前述したような構文を取る「MAKE」の機械翻訳について報告する。

1．SRLによる辞書記述

この機械翻訳システムでは、意味表現用言語SRL[2]を基に、今回これを翻訳用辞書の記述に適した記法を導入している。この新しい記法を用いた「TAKE」の辞書項目記述（図1）について説明する。

図1(a)は「TAKE」という見出しを持つ単語の品詞がV（動詞）であることを記述している。(b)は「TAKE」がVT（他動詞）かつVI（自動詞）であり、ホーンピー文型パターン[3]のVP6A, VP15A, …… VP22を取り得ることを示している。これらは「TAKE」のメッセージとして主として構文解析時に使う。(c)は「TAKE」のデフォールト訳語を「取る」としている。

図1(d)は目的語が「VEHICLE」の時は「～に乗る」と訳せ。という意味である。詳しく言うと、動詞「TAKE」のobjスロットの値が「VEHICLE」であれば、次に来るLisp関数を実行する。関数(@HONYAKU * NI 乗R.U)を実行すると、目的語の訳語はそのままにして（「*」は訳語をそのまま当てはめるという意味）、それに助詞「NI（に）」を付加する。そして最後に「TAKE」のデフォールト訳語「取る」を「乗る」に変える。その後、次の関数@WAIT-AND-SEEを実行する。この関数は引数であるwith… 以下をデモンとして保存しておき、他の処理が終わった後で実行する。最後の「=goal1」はスロット名をobjからgoal1に変えることを意味する。

図1(f)は「TAKE」が「WITH」の率いる前置詞句を伴う時、「WITH」の目的語が「HUMAN」ならばその助詞を「～とともに」とし、そうでなければ助詞を「～をもって」とすることを示す。そしてスロットの名称をwithスロットとする。また(g)は「TO」の率いる前置句を伴う時、その目的語が場所を表す（そのsemantic featureがlocationである）ならば「～へ」と訳し、物体ならば「～のところへ」

と訳すことを意味する。また「=goal」によってスロット名をgoalに変える。

ここで次の文を例に取り、@WAIT-AND-SEEについて説明する。

(1) I take a plane with a gun to Moscow.

(2) I take a plane to Moscow.

「VEHICLE」は「PLANE」の上位概念であるので、前述した図1(d)の処理を行って、「TAKE」のgoal1スロットの値は「PLANE」となる。この時はまだ前置詞句の処理(図1(f),(g))を行っていないので、前置詞句に関する情報は得られない。そのため、「WITH」,「TO」などの前置詞句がある場合に、どういう翻訳をするかというプログラムをデモンとして残すのが関数@WAIT-AND-SEEの働きである。目的語に続いて前述のように前置詞の処理を行い、例文(1)では「TAKE」の各スロットに次のような値が入る。

goal1 = 「PLANE (飛行機に)」
with = 「GUN (銃をもって)」
goal = 「MOSCOW (モスクワへ)」

このように、単文の処理が一通り終わった段階では、それぞれの語句はすでにスロットの値となっているので、ここで初めて@WAIT-AND-SEEで作ったデモンを起動させる。@WAIT-AND-SEEの引数(デモン)は(e1)~(e5)及び(e6)~(e7)の2個である。(e1)はwithスロットの値が「WEAPON」のとき、以下の(e2)~(e5)を実行せよ。

```
(a)  (TAKE V
(b)       ((VT VP6A VP15A VP15B VP2A VP12A
            VP13A VP14 VP16B VP2A VP14 VP22) U)
(c)     '((TAKE  取 R . U)
          unit
          (part-of)
          (self (a DEFAULT-CASE))
          (sf !+action)
          (subj
             (((OR (sf pronoun) HUMAN))
                =agent))
          (iobj
             ((ANIMAL)
                (@HONYAKU * NOTOKOROHE *)
                =goal))
          (obj
(d)          ((VEHICLE)
                (@HONYAKU * NI 乗 R .U)
(e)             (@WAIT-AND-SEE
(e1)               (with ((WEAPON)
(e2)                  (@HONYAKU1 * DE 乗っ取 R . U)
(e3)                  (@HONYAKU2 goal1 WO *)
(e4)                  (@WAIT-AND-SEE
(e5)                     (goal ((T)
                            (@HONYAKU1 * * 乗っ取って行 K . U))))))
(e6)               (goal ((T)
(e7)                  (@HONYAKU1 * * 乗 TTEIK . U))))
                =goal1)
        )           中略

(f)     (prep
           ((HUMAN WITH)
              (@HONYAKU * TOTOMONI * ))
           ((T WITH)
              (@HONYAKU * WOMOTTE *))
           =with)
(g)     (prep
           (((sf location) TO)
              (@HONYAKU * HE *))
           ((OBJECT TO)
              (@HONYAKU * NOTOKOROHE *))
           =goal)
        ))          後略
```

図1 TAKEの辞書記述

という意味である。例文(1)ではwithスロットの値は「GUN」であるので、以下を実行する(「WEAPON」は「GUN」の上位概念)。(e2)の関数@HONYAKU1では、「銃」に付いた助詞を「で」に変え、「TAKE」の訳語を「乗っ取る」とする。(e3)の関数@HONYAKU2ではgoal1スロットの値「飛行機に」の助詞を「を」に変える。(e4)で再び@WAIT-AND-SEEを実行し、デモン(e5)を作る。

ひとつの@WAIT-AND-SEEで作られるデモンは、COND文のように択一選択で実行する。つまり上のように1つめのデモン(e1)~(e5)が実行された時は2つめの(e6)~(e7)は実行しない。

@WAIT-AND-SEEが(e)のようにネスト構造のとき、内側で作られたデモン(e5)は外側のデモン(e1)~(e5)の実行終了後すぐに起動される。(e5)ではgoalスロットが存在するとき(「T」は存在すれば良いことを示す)、次の関数@HONYAKU1を実行し、「乗っ取って行く」と訳す。

例文(2)について考えると、withスロットはないので1つめのデモンは実行しないで2つめの(e6)~(e7)を実行する。ここでは「乗っていく」と訳す。このようにCOND文のような構造で複数のデモンを残せるため、排反な事象を記述しやすい。

@WAIT-AND-SEEを使ったこのような辞書記述は語意トランスファ用の規則がコンパイルされている。そして意味解析の過程において、その時点での最善の訳語を選択するとともに、さらにどのような語意トランスファ規則を用いるかが絞られていく。また、一通りの処理が終わった段階でデモンを起動させるため、各スロットの値となる語句の順序に影響されないという特徴がある。

2. 文法規則

文法規則は拡張LINGOL[4]で使われるn進木補強文脈自由文法規則を用いている。図2に例として基本的な平叙文を解析するための文法規則の記述を示す。図2(a)の記述は SDEC → SUBJ (AUXD) (ADV) V (OBJ1) (OBJ2) に対応する。AUXD, ADV, OBJ1, OBJ2の前についた@印はそれぞれが省略可能なことを意味している。それぞれの非終端記号の後に付加されたのが、<advice>部のプログラムである。これは構文解析時に実行され、その結果がNILの時に限り、対応した文法規則を適用可能とする。図2(b)は<cog>部のプログラムで、この文法規則の優先順位を表す。[6] ここには通常数値が入る。

図2(c)は意味解析のための<sem>部のプログラムである。<sem>部のプログラムは構文解析が終了した時点で実行される。関数@SDECの引数は、SUBJ, V, OBJ1, OBJ2などの意味解析結果及びVに関するメッセージである。最後の'(SUBJ OBJ1 OBJ2 V)はこれらの記号に対応した語がこの順序で並ぶという日本語の語順を指定している。

```
(a) (SDEC ( (SUBJ NIL)
            @(AUXD (PMB (MMB)))
            @(ADV NIL)
             (V (@CONCORD (PMB 'SUBJ) (PMB 'AUXD)))
            @(OBJ1 (COND ((NULL (MEMQ 'VT (PMB 'V))))
                         (T (PMB (SETAND '(EMBED) (MMB))))))
            @(OBJ2 (COND ((NULL (PMB 'OBJ1)))
                         ((MEMQ 'EMBED (PMB 'OBJ1)))
                         ((@OBJ2? (PMB 'OBJ2)))
                         ((MEMOR '(VP12A VP12B VP12C) (PMB 'V))
                          (PMB (SETAND '(EMBED) (MMB))))
                         (T T))))
(b)      0
(c)      (@SDEC (INTERP 'SUBJ)
                (@TR-FORM (@COPY-U (INTERP 'V)))
                (INTERP 'OBJ1)
                (INTERP 'OBJ2)
                (PMB 'V)
                '(SUBJ OBJ1 OBJ2 V)))
```

図2 文法規則記述例

関数@SDECでは動詞(V)のsubjスロットに主語(SUBJ)の意味解析結果、objスロットに目的語(OBJ1)の意味解析結果を代入する。また二重目的語があるときは、動詞(V)のiobjスロットに間接目的語(OBJ1)、objスロットに直接目的語(OBJ2)の意味解析結果を代入する。

3.「MAKE」の訳し分け

図3に「MAKE」の辞書項目記述を示す。これを用いて、「MAKE」の訳語選択について説明する。

3.1 目的語による訳し分け

3.1.1 間接目的語の限定

図3(d)は「MAKE」のiobjスロットの値(間接目的語)が「ANIMAL」または代名詞のとき次の関数を実行せよ、ということを意味する。@HONYAKUを実行すると、「～のために」という助詞を付ける。「=goal」により、このスロット名をiobjからgoalに変える。iobjスロットの値が「ANIMAL」でも代名詞でもないときは、その構文解析木に対する意味解析に失敗し、翻訳結果を出さない。つまり間接目的語は動物に限定する。しかしiobjスロットに当てはまる値がない時は、以下の処理を行う。

3.1.2 目的語による訳語の決定

図3(e1)は「make a trip」のように目的語によって単純に「MAKE」の訳語が定まる場合、objスロットの値となる単語の辞書項目中に訳語が記述されていることを意味する。関数@IDIOMは目的語のidiom1スロットに「MAKE」を代入する。図4に例として「TRIP」の辞書項目記述を示す。ここで(a)

```
(a) (MAKE V
(b)      ((VI VT VP6A VP14 VP12B VP13B VP16A VP22
           VP24A VP2A VP18B VP23A VP23B VP25
           VP12A VP13A VP2C VP3A VP15B OF) 0)
(c)    '((MAKE 作 R . U)
         unit
         (part-of)
         (self (a DEFAULT-CASE))
         (sf !+action)
         (subj
            (((OR (sf pronoun) HUMAN))
               =agent))
(d)      (iobj
            (((OR (sf pronoun) ANIMAL))
               (@HONYAKU * NOTAMENI *)
               =goal))
         (obj
(e1)        (((TO-FILL (@IDIOM1))))
(e2)        ((ANIMAL)
               (@HONYAKU * WO *)
               (@WAIT-AND-SEE
                  (goal ((T)
(e3)                 (ERR 20 NIL)))))
            ((T)
               (@HONYAKU * WO *)))
(f1)     (proposition
(f2)        ((T)
(f3)           (@SHIEKI FILLER)
               (@WAIT-AND-SEE
                  (obj
                     ((ANIMAL)
                        (@HONYAKU2 * NI NIL SE . RU))
                     ((T)
                        (@HONYAKU2 * WO NIL SE . RU))))))
         (comp
(g1)        ((ANIMAL)
               (@HONYAKU * NI NIL S . URU))
(g2)        ((T)
               (@HONYAKU * NI NIL S . URU)
               (@WAIT-AND-SEE
                  (obj ((ANIMAL)
                     (ERR 20 NIL))))))
         (acomp
(h1)        ((SURE)
               (@HONYAKU NIL * 確 KAME . RU)
               (@WAIT-AND-SEE
                  (of ((T)
                     (@HONYAKU1 * WO *)))))
(h2)        ((READY)
               (@HONYAKU NIL * 用意 S . URU)
               (@WAIT-AND-SEE
                  (for ((T)
                     (@HONYAKU1 * WO *)))))
(h3)        ((T)
               (@SHIEKI FILLER)
               (@HONYAKU * * NIL S . URU)))
         (prep
(i1)        ((T FROM)
               (@HONYAKU * KARA *)
               =from)
(i2)        ((T OF)
               (@HONYAKU * DE *)
               =of)
(i3)        ((T INTO)
               (@HONYAKU * NI 作 RIKAE . RU)
               =into))
         (prep
(j1)        (((OR (sf pronoun) ANIMAL) FOR)
               (@HONYAKU * NOTAMENI *)
               =goal)
(j2)        ((T FOR)
               (@HONYAKU * NOTAMENI *)
               =for))
         (prep
(k1)        ((T OUT)
               (@HONYAKU * WO 理解 S . URU))
(k2)        ((FACE UP)
               (@HONYAKU * WO 化粧 S . URU))
(k3)        ((MIND UP)
               (@HONYAKU NIL * 心を決 ME . RU))
(k4)        ((T UP)
               (@HONYAKU * WO 作り上 GE . RU))
(k5)        =obj)))
```

図3 MAKEの辞書項目記述

はidiom1スロットに代入される値が「MAKE」なら次の関数を実行せよ、という意味である。関数@RHONYAKUは、「TRIP」の訳語「旅行」に助詞「を」を付加し、idiom1スロットの値「MAKE」の訳語を「する」に変える。

```
(TRIP N ((C) 0)
        '((TRIP 旅行)
          unit
          (part-of)
          (self (a ACTIVITY))
          (sf !+action)
(Q)       (idiom1
             ((MAKE)
                (@RHONYAKU * WO NIL S . URU)))))
```

図4　TRIPの辞書項目記述

このように動詞の訳語を名詞に書くことによって、動詞の辞書項目だけが巨大化することを防ぐことができる。名詞の方も上位概念を使うことにより、目的語となり得るすべての名詞の辞書項目中に記述する必要はない。例えば、名詞「LIQUID」の辞書項目中に、これを目的語とする「TAKE」の訳語を「飲む」と記述すれば、take coffee, take milk など、「LIQUID」を上位概念として持つ名詞（「COFFEE」,「MILK」）には記述しなくとも「TAKE」の訳語は「飲む」となる。

「MAKE」が上のようにうまく目的語のidiom1スロットに当てはまったときは、以下の(e2)～(e4)は行わない。

3.1.3　目的語と名詞補語の識別

英文がSUBJ V NP1 NP2 という並びを持つとき、①SUBJ V OBJ1 OBJ2 及び②SUBJ V OBJ1 COMP という2つの構文解析木が作られる可能性がある。このシステムでは、ホーンビーの文型パターンを用いて、①二重目的語を取り得る動詞(VP12A,B,C)　又は　②目的語と名詞の補語を取り得る動詞(VP23A,B)のチェックを行っている。しかし「MAKE」のように①②両方が可能な動詞(図3(b))では、2つの構文解析木が作られる。ここではNPが動物であるかどうかに着目してNP1が動物でNP2は動物でない時に限り二重目的語（例. I make him a desk）、それ以外のときは名詞補語(例 I make him a doctor.)として翻訳を行う。

図2(e2)では、obj スロットの値が「ANIMAL」のとき、以下の関数を実行する。まず@HONYAKUによって助詞「を」を付加する。次の関数@WAIT-AND-SEEでは、goalスロットが値を持つならば関数(ERR 20 NIL)を実行するデモンを作る。①の構文解析木について意味解析を行っているとき、3.1.1で述べたように、間接目的語が動物ならばgoal スロットにその値が代入される。そのためデモンを実行することによって関数ERRを実行し、構文解析木①に対する意味解析に失敗する。つまりNP1, NP2 共に動物のときは二重目的語とする翻訳結果は出さない。

3.1.4　その他の目的語による翻訳

「MAKE」の目的語が図3(e1)(e2)に当てはまらない時は、(e3)を行う。(T)はobj スロットはどんなものであっても以下の関数を実行することを示す。ここではデフォールト訳語「作る」を取る。

3.2　便役の意味を持つ「MAKE」の翻訳

3.2.1　文法規則記述

不定詞を持つ平叙文を解析するための文脈自由文法規則は次の通り。

SDEC → SUBJ (AUXD) (ADV) V (OBJ1) INFINITIVE

この文法規則に対応する意味解析のプログラム〈sem〉部では、不定詞がTOを伴わない時は次のことを行う。SUBJの意味解析結果をVのsubjスロットに代入する。OBJ1の意味解析結果をVのobjスロット及びINFINITIVEのsubjスロットに代入する。INFINITIVEの意味解析結果をVのpropositionスロットに代入する。

3.2.2 辞書記述

不定詞を伴った使役の「MAKE」のための辞書記述は図3(f)の部分である。(f1)はpropositionスロットに値が代入されたなら、つまり「MAKE」が原形不定詞を伴うとき、(f2)(f3)を実行せよ、という意味である。(f2)の関数@SHIEKIは原形不定詞の訳語を次のように「せる」が付く活用形にする。

五段活用動詞	行K.U →	行K.A
上一段活用動詞	見.RU →	見.SA
下一段活用動詞	得.RU →	得.SA
カ変活用動詞	来.RU →	来.SA
サ変活用動詞	S.URU →	S.A

＋「せる」

(f3)の関数@WAIT-AND-SEEでは、目的語によって助詞「に」と「を」の選択を行っている。目的語が動物のとき、「～に…せる」と訳し、そうでないとき「～を…せる」と訳す。 例 I make him go.(彼に行かせる)
I make the egg addle.(卵を腐らせる)

関数@HONYAKU2は目的語の助詞を付け変え、「MAKE」の訳語を「せる」にしている。

付録1にI make him go.の意味構造抽出結果を示す。これは翻訳された日本語の語順に従って、対応する単語のユニットを並べたものである。単語のとなりに書かれた日本語を順番に並べると翻訳結果となる。「HE」が「MAKE」のobjスロットと同時に「GO」のagentスロットの値となっていることがわかる。

3.3 名詞補語を伴った「MAKE」の翻訳

I make him a doctor.のように、目的語及び名詞補語の両方を持つ文を解析するための文法規則は SDEC → SUBJ (AUXD) (ADV) V OBJ1 COMP である。この文法規則に対応する意味解析のプログラム〈sem〉部では、SUBJ, OBJ1, COMPの意味解析結果をそれぞれVのsubj, obj, compスロットに代入する。

辞書記述図3(g1)は名詞補語が動物であるとき、「～に…する」と訳す。(g2)は名詞補語が動物でないとき、「～に…する」と訳した後デモンを作る。これは、目的語が動物のとき意味解析に失敗する。I make him a desk.のような文をこの構文解析木を持つものとして解析すると、名詞補語(desk)は動物でなく、目的語(him)は動物なので意味解析に失敗する。(3.1.3 参照)

3.4 形容詞補語を伴った「MAKE」の翻訳

I make him happy.のように形容詞補語を持つ文を解析するための文法規則は、SDEC → SUBJ (AUXD) (ADV) V (OBJ1) ADJP である。この規則に対応する〈sem〉部では、SUBJ, OBJ, ADJPの意味解析結果をそれぞれVのsubj, obj, acompスロットに代入する。

辞書記述図2(h1)は形容詞補語が「SURE」のとき、「SURE」の訳語をNILとして(「SURE」は訳さない)、「MAKE」の訳語を「確かめる」としている。また@WAIT-AND-SEEにより、ofスロットに値が代入されればその訳語の助詞を「を」に変える。ofスロットについては後に3.5.1で述べる。

(h2)は形容詞補語が「READY」のとき、上と同様に、「MAKE」の訳を「用意する」とし、forスロットに値が代入されれば、その助詞を「を」と変える。

(h3)は形容詞補語が「SURE」「READY」以外のとき、@SHIEKIを実行し、形容詞の訳語を助動詞「する」が付く活用形にする。

形容詞	美SI .I	→	美SI .KU	+「する」
形容動詞	幸福N.A	→	幸福 .NI	
名詞+「の」	緑色 .NO	→	緑色 .NI	

その後@HONYAKUを実行し、「MAKE」の訳を「する」に変える。ADJの訳となる日本語が上の3つ以外(DRUNK(酔った)など)については検討中である。

3.5 前置詞を伴った「MAKE」の翻訳

3.5.1 前置詞「FROM」「OF」「INTO」を伴った「MAKE」の翻訳

図2(i1)(i3)によって「MAKE」が前置詞「FROM」「INTO」を伴うとき、「~から作る」,「~に作りかえる」と訳す。その後スロット名をfrom, intoに変更する。

(i2)では、前置詞「OF」があれば、「~で作る」と訳す。その後スロット名をofに変更する。これにより、(h1)でデモンを作っておれば、これを起動した時に@HONYAKUを実行する。

前置詞は動詞にかかるときと名詞にかかるときがあるので、前置詞句を持つ文は、それだけで2つの構文解析木を得る。しかし「OF」は名詞にかかることが多いために、普通は動詞にかかるような構文解析木を作らない。「MAKE」のように「OF」がかかり得る動詞には、メッセージに「OF」を加える(図3(b))。構文解析時には、「OF」をメッセージとして持つ動詞のときに限り解析木を2つ作り、それぞれについて翻訳を行う。「OF」以外の前置詞は、「MAKE」における「FROM」などのように、動詞のprepスロットを満たし得る時は、名詞にかかるような構文解析木に対する翻訳を行わない。

3.5.2 前置詞「FOR」を伴う「MAKE」の翻訳

図2(j)は前置詞「FOR」を伴うとき、「~のために」と訳す。このとき「FOR」の目的語が動物ならば、それをgoalスロットとする。これによって、I make him a desk. とI make a desk for him. から同じ深層構造を得られる。「FOR」の目的語が動物でないとき、スロット名をforにする。

3.5.3 「MAKE」が作る二語動詞の翻訳

図2(k1)	「MAKE OUT」	→	「~を理解する」
(k2)	「MAKE UP FACE」	→	「顔を化粧する」
(k3)	「MAKE UP MIND」	→	「心を決める」
(k4)	「MAKE UP ~」	→	「~を作り上げる」

と翻訳し、(k5)でスロットの名称をprepからobjに変える。

あとがき

「MAKE」の作るほとんどの構文について翻訳できるようになった。それらの翻訳例を付録2に示す。「MAKE」に関する訳し分けはとりあえずこのくらいにして現在は「GET」について行っている。「GET」はI get the machine to go.を「機械を動くようにする」という解釈の他に「動くべき機械を得る」という解釈が出てしまうというようなあいまいさが多い。しかし、他の点では現在の辞書記述方式で書けると思われる。また次には「HAVE」について行う予定である。

辞書の記述方式は現在の方法におおむね決定する予定である。しかし、よりユーザにわかりやすい辞書記述の開発及びそれから現在の方式へのトランスレータの製作は今後の重要な課題である。

謝辞

計算機使用に関し、便宜を計っていただいた渕一博氏を始めとするICOTの皆様に感謝致します。また熱心に討論していただいた東工大田中研究室の皆様に感謝致します。

参考文献

1) 田中穂積：「解析から合成までを融合した英日機械翻訳システム」，日経エレクトロニクス，pp.275-293，1983年8月29日

2) 田中，元吉，安川：「意味表現用言語SRLの機械翻訳への応用」，情報処理学会自然言語研究会資料，31-5，1982年5月

3) ホーンビー：「英語の型と語法」，オックスフォード大学出版局，1981年

4) 前見達夫：「LINGOLのn進木への拡張」，東京工業大学総合理工学研究科修士論文，1980年3月

付録1 I make him go. の意味構造抽出結果

```
$ I MAKE HIM GO.

WAKATTAYO!!

READY?!
$

1 PARSING TREES
$
!
READY?!
$

意味構造

$

'*************************'

((((I 私 . HA) 5)
   unit
   (part-of)
   (self (a HUMAN))
   (sf pronoun))

'*************************'$

'*************************'

(((HE_1 彼 . NI) 7)
   unit
   (part-of)
   (self (a HUMAN))
   (sf pronoun))

'*************************'$

'*************************'

(((GO 行 K . A) 8)
   unit
   (part-of)
   (self (a DEFAULT-CASE))
   (sf faction)
   (agent
      =
      ((HE_1 彼 . WO) 7)))

'*************************'$


'*************************'

(((MAKE NIL SE . RU) 6)
   unit
   (part-of)
   (self (a DEFAULT-CASE))
   (sf faction)
   (agent = ((I 私 . HA) 5))
   (obj
      =
      ((HE_1 彼 . NI) 7))
   (proposition
      =
      ((GO 行 K . A) 8)))

'*************************'$

翻訳結果

私は彼に行かせる。
```

付録2、「MAKE」の訳し分け例

```
I MAKE A SHORT TRIP.

WAKATTAYO!!

1 PARSING TREES
!

翻訳結果

私に短かい旅行をする。


I MAKE HIM A DOCTOR.

WAKATTAYO!!

2 PARSING TREES
!

翻訳結果

私は彼を医者にする。
```

```
I MAKE HIM HAPPY.

WAKATTAYO!!

1 PARSING TREES
!

翻訳結果

私は彼を幸福にする。


I MAKE HIM A DESK.

WAKATTAYO!!

2 PARSING TREES
!
!

翻訳結果

私は彼のために机を作る。
```

下の2つは構文解析木を2つ得るが，片方の木の意味解析に失敗し，翻訳結果がでない。