TR-I-0062

対話文翻訳における英文生成システムの検討

An Experimental Study on English Sentence Generation System for the Natural Dialogue Translation

上田良寛
Yoshihiro Ueda

1988.12

概要

英文生成に用いる文法･辞書を、英文解析のそれと共有させる双方向性のある生成システムを実現する方法について述べている。このシステムは素性構造を入力とし、メカニズムとしてユニフィケーションを採用している。双方向システムは、コントロールを明示的にしなければならない木構造書き換えシステムと比較して、文法･辞書の開発が容易になる、解析文法の不備をみつける検証ツールとして利用可能であるなどの利点がある。また意図伝達方式との整合性もよい。

書き換え規則の選択のために、適用される素性構造を指定することができるようにし、効率の向上を目指している。

いくつかの英語文法を作成し、比較実験を行った。さらに、日本語生成実験も行った。これから得られた結果をもとに、効率の向上や文法の強化のために行わねばならない課題を考察した。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

目 次

1 背景と目的

ATR自動翻訳電話研究所では、自動翻訳電話の前段階として、端末間通信の翻訳システムNadine (Version 1)を開発した。ここでは、解析部に重点がおかれ、パーサおよびかなり詳細な日本語文法が作成された。変換部と生成部に関しては既存の技術が利用されていた。これは、コントロールを明示的に示した、木構造書き換え規則によるものである。

Version 2以降では、変換部と生成部に関する本格的検討が必要になる。生成部においては、次のような課題がある。

1) Intentionの反映

話し言葉においては、発話意図は重要な意味をもっており、翻訳結果にこれが正しく反映されなければならない。このため、ATRでは意図伝達方式を研究の一つの柱としており、解析の結果として得られる素性構造には、発話内容に加え、発話意図も同時に含まれている。これが変換部をとおして生成部に渡されるので、発話意図を含めた生成を行うための情報は十分に用意されていることになる。

しかし、コントロールを明示した木構造書き換えでは、この情報は書き換え規則の中に埋もれてしまい、発話意図と表層文の関係が明らかにされない。

2) 文法記述の容易さ

文法規則および辞書が複雑になるのを避けるため、言語解析においては種々の文法理論が採用されてきている。生成においても、文法規則および辞書が複雑になることは同様であるため、文法記述が容易にできるように生成部を設計することが重要である。

現在の方法では、文法の記述の際に、文法全体の制御構造を把握しておかなければならず、容易とは言えない。また、文法の一貫性を保つためには、解析用文法を変更した場合には、生成用文法においても同じ変更を行わねばならない。

これらの課題を満足させるため、宣言的な文法記述を可能にし、さらに解析部と文法規則/辞書を共有可能にした生成システムを考える。これの基礎となるメカニズムには、解析部と同様にユニフィケーションを採用する。この報告では、この方針で開発した予備実験用生成システムについて報告する。

第2章では、双方向文法生成システムの概念について説明し、予備実験用生成システムの設計目標について述べる。第3章では、このシステムにおけるコントロールメカニズムについて説明する。第4章で、生成システムに与える文法の記述方法について述べ、それにしたがって作成された文法(日本語、英語)の説明を第5章で行う。これらによる生成結果を第6章に示す。第7章と第8章では今後の課題をあげる。第7章では、比較的短期的な課題であるこの予備実験用生成システムの改良点に付いて述べ、第8章では、生成の研究で行わねばならない長期的な課題について言及する。

2 設計目標

ここでは、このシステムの設計目標を述べる。まず、そのうちもっとも重視している双方向性について説明する。

2.1双方向システム

2.1.1 双方向的文法

ここで考える生成システムは、解析の逆のプロセスをユニフィケーションという同じメカニズムで処理しようというものである。解析の場合、個々の辞書エントリに与えられた素性構造が、CFG規則に与えられた制約(実際は素性構造の形式をしている)にしたがってユニフィケーションが行われ、最終的に解析される文に対しての素性構造が与えられる。生成の場合は、文に対して与えられた素性構造を、CFG規則に与えられた素性構造とユニフィケーションを行い、文に対する句構造を得ることを目的としている。

このような生成システムでは、文法、辞書は、解析にも生成にも使うことができる。このような文法を双方向的文法という。

双方向的ということは、ある文法を用いて得られる解析結果(ここでは素性構造)を生成システムに与えると、同じ解析結果が得られる表層文を出力するということである。文法によっては、複数の表層文を解析した結果が1つの同じ素性構造になる場合がある。この場合は、複数の生成結果が得られることになる。このことは、欠点にもみえるが、次に述べる理由により、利点ということもできる。

2.1.2 双方向性の利点

双方向性の利点としては、次の3つがあげられる。

1) 文法、辞書のメインテナンス

双方向であることにより、文法の記述量を減らす。解析と生成で別々の文法記述を行わねばならない場合、一方の文法を変更したら、もう一方の文法も同様の変更をしなければならない。双方向性は、そのような文法の二重管理を避けることができる。なお、解析と生成では文法に要求される事項が異なるので、単純に記述量が半分になる訳ではない。重要なことは、解析と生成で文法の一貫性が保たれることである。[1]

なお、双方向であるためには、文法は純粋に宣言的でなければならず、手続的な部分は排除しなければならない。

1解析では入力される文が正しいと仮定したときには、文法にそれほどの厳密さを要求されない場合がある。例えば、主語と述語動詞の一致などがある。しかしこれにしても、ある動詞に対する主語があいまいであるときなどのdisambiguationに用いることができる。この意味では、「厳密さを要求されない」ということは一概には言えないことになる。

2) 解析文法の検証ツールとして

解析文法はあまり厳密でなくとも正しく処理できる可能性がある(脚注1参照)が、生成の場合にこれをそのまま用いると、非文を生成してしまう。入力される文が正しいと仮定されていて、解析文法にさほどの厳密さが要求されないならばよいが、厳密な文法が望まれている場合には問題がある。

このようなとき、このような生成システムは正しい文法を記述するためのツールとして使うことができる。生成結果が複数得られ、その中に非文が混じっている場合に、文法に誤りがあることになるのでチェックを行わねばならない。このような生成システムがなければ、種々の非文をテストデータとして作り、解析システムにかけてみて正しく排除されるかを調べる必要がある。

3) 生成および翻訳の研究の基本ツールとして
— 生成に必要な情報、解析で抽出しなければならない情報は何か

文法的には正しくとも、複数の結果がえられる場合がある。この中には、もとの文に対する翻訳としては適切でないものも含まれることがある。異なった表層文が同じ意味構造に対応していても、発話の効果が同じとは限らない。これらの違いが適切に意味構造に反映されていれば、訳し分けも可能になる。

このように結果を検討していくことにより、「生成に必要な情報は何か」、さらには、「原言語の解析の際に、付け加えておかなければならない情報は何か」が分かる。コントロールを明示的に行う生成の場合、このような情報が意味構造の中に入っていなくとも、「原文にとって適切な」よい文を生成するためにコントロールで工夫することができる。その結果として、必要な情報が埋めこまれてしまう恐れがある。このように、コントロールを文法記述から独立させて、生成結果を詳細に検討することは、対話文の生成の研究をする上においては不可欠と言える。

2.1.3 双方向性の欠点とその解決

一方、これに対して、文法的な知識は共有できても、人間の処理メカニズムは全く異なるため、処理メカニズムおよびそれを駆動するための知識は異なったものが必要になるという意見がある(McKeown 1985)。しかし、発話行為(生成)は、聞き手がどのような認識(解析)をし、どのような効果を生むかを計画して行われていると考えることもできる。発話意図も、相手に対する効果を考慮して、表層の発話の中に組み込まれる。表層の発話は、発話意図も含めた情報がエンコードされたもので、それを聴者が適切にデコードしてくれるのを期待されているのである。ここで問題になるのは、表層の発話と、それが担う意味の関係だけであって、解析および生成の処理メカニズムは問題にしなくてもよい。意図伝達方式は、発話意図を含めた情報のエンコーディングを目指すものであり、このような考え方の基礎を与えるものである。

双方向性の欠点として、Appelt (1987)は"logical-form equivalence"の問題をあげている。これは、意味表現である論理形式では、同じ意味を表すのに複数の表現方法があり、これらは異なった表層文として現れ、その意味も異なってくるというものである。これを避けるために、通常はプランニングの段階で暗黙に表層の文法を考慮して論理形式を組み立てる。しかし、意図伝達方式を採用すれば、意味的に異なる表層文の違いは、意味表現の中で(おそらくは発話意図の違い

として)適切に表現されるし、素性構造による中間表現は一意に決定できるので、この問題は解決される。

2.2設計方針

ここでは、生成システムの基本的な設計方針として、双方向性をもたせること(設計方針1)以外のものを列挙する。

解析過程は、次の2つのステージで行われることが想定されている(Kogure, 1988)。

1) Unification-based syntactico-semantic analysis
2) Speech act plan recognition inference

これに対応する生成過程も、同様に2つのステージで行われることになる。ここで考える予備実験用生成システムは、このうち、(1)に対応する部分を受け持つ。単文レベルの解析に対応する、単文レベルの生成を行うところで、入力には表層上の発話意図が与えられていることを想定している(方針2)。また、これは、テキストプランニングは行わないことを意味している。

生成の対象となる言語は、英語を主に考える(方針3)。しかし、メカニズム自体は言語によらないものとし、日本語での生成の実験も行う。

3 生成メカニズム

ここでは、予備実験用生成システムの生成メカニズムについて記述する。まず、ここで用いられているデータ構造について説明し、3.3で手順をまとめる。

3.1 c構造

書き換え規則を逆に適用した結果を残すために、素性構造を共有する木を表現するデータ構造を導入する。この木構造をc構造(constituent structure)と呼ぶ。このノードには、以下のような情報が記載される。

cat	カテゴリ(書き換え規則のラベル)
fs	素性構造
mother	親ノード
seq	子ノードのリスト

例えば、図のようなc構造を考えることができる。この図ではfsそのものが共有されているように描かれているが、fsの一部(いずれかの素性の値)が共有されていてもよい。ここで示されている素性構造も、実際の素性構造の中の意味素性だけを抜き出したものになっている。

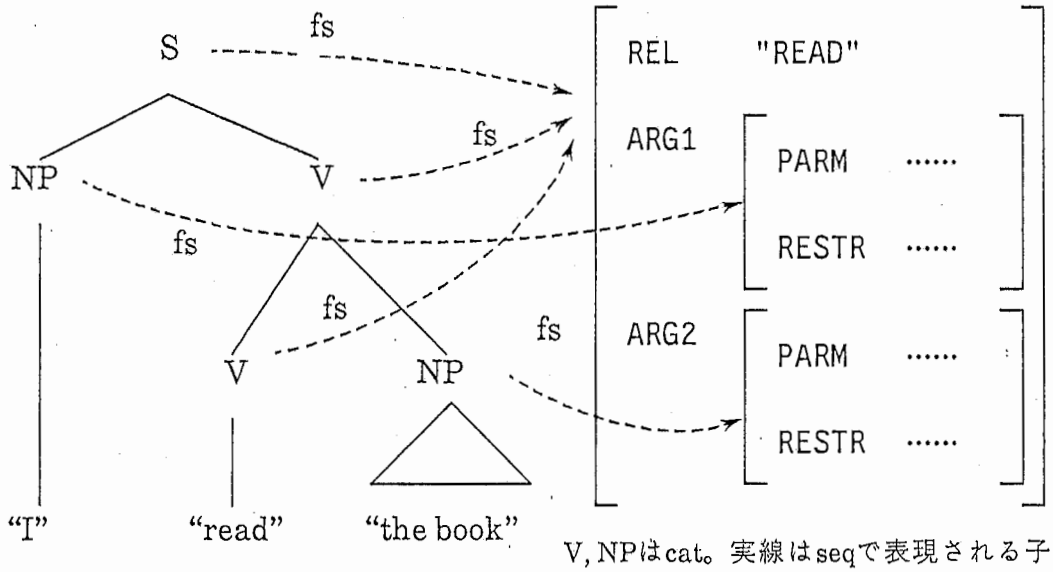

図3-1 c構造と素性構造

生成は、トップレベルの素性構造をもつノードから、書き換え規則を解析の逆に適用していき、素性構造をを共有するc構造をトップダウに組み立てていくことにより達成される。最終的なc構造からターミナルノードを順に集めていくことにより、表層文が得られる。

3.2ペンディングノードリスト

生成に入力される素性構造には、意味素性と文の時制などのいくつかの素性だけが記述されていると仮定される。主として意味素性が生成に用いられる。書き換え規則を逆に適用した後に、意味情報が適切に分配されないと、その後の書き換えは不可能になる。例えば、動詞とSUBCATの要素(ここでは目的語のNPを仮定する)を対応づける規則

```
V -> V X
   <0 TRANS> == <1 TRANS>
   <1 SYNCAT REST FIRST> == <2>
```

には、X(NP)に対するTRANS素性(意味素性)の指定はない。これは、X全体がVのSYNCAT(一般にいうSUBCATのこと)に入り、このVの中で詳細な記述がなされているからである(実際は語いまで下がったところに記述されている)。

この場合、Vのほうの書換えが進み、語いまで書き換えた段階で初めて、X(NP)にTRANS素性が供給される。したがって、Xのほうの書き換えは、それまで待たねばならない。このような待ち状態にあるノードを保持するものとして、生成結果に対応するツリーごとにペンディングノードリストを用意している。また、書き換え規則が適用できるツリーはペンディングツリーリストに保持されている。

c構造のデータ構造は次のフィールドを含んでいる。

root	c構造のルートに対するポインタ
pending-nodes	ペンディングノードリスト
processed-nodes	処理ずみのノードのリスト (ペンディングノードリストに入っていないノード)

書き換え規則の適用はペンディングツリーリストから1つのツリーを選択し、そのペンディングノードリストの中から1つのノードを選択することになる。このように、生成における書き換えは、横型で進められていることになる。この機構にアジェンダを組み合せることで最適なコントロールを行う可能性があるが、現在のところはアジェンダは組み込んでいない。

3.3生成手順まとめ

生成手順をまとめると、次のとおりになる。

1) ペンディングノードのうち意味素性の付けられているものを選択し、そのノードを含むc構造をペンディングツリーリストから取り除く。

2) そのノードのCATを左辺にもつ書き換え規則のうち、選択したノードと規則とユニフィケーションを行って成功したものを選ぶ。ユニフィケーションの結果得られたノードの子ノードをそのc構造のペンディングノードリストに追加し、そのc構造をペンディングツリーリストに追加する。

3) 適用可能なペンディングノードがなくなるまでこの操作を繰り返し、すべてのリーフがターミナルになっているc構造を表層文生成ルーチンへ渡す。

現在の手順では、ターミナルに対する書き換えも全てユニフィケーションを行ってチェックしている。ここも改善の必要がある。

図3-2の素性構造を入力とした場合の例で動きを示す(図3-3)。

```
[[HEAD [[FORM FINITE]
        [TENSE PAST]
        [MODE DECLARATIVE]]]
 [TRANS [[RELATION PERFECTIVE]
        [ARG1 [[RELATION "SLEEP"]
               [ARG1 [("ARTHUR"の意味素性)]]]]]]]
```

図3-2 入力素性構造

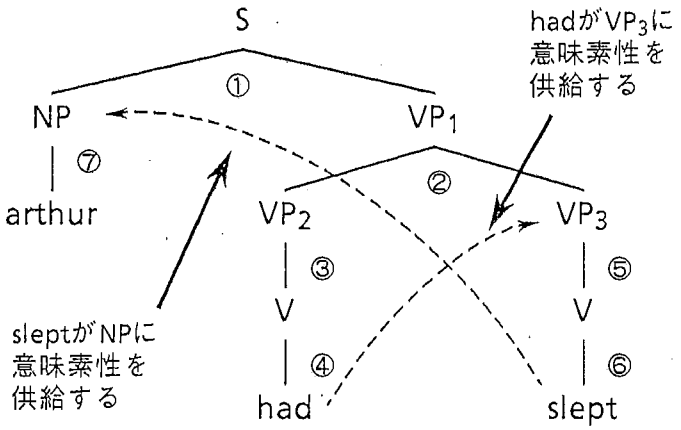

図3-3 生成例 ("arthur had slept")

1) 図3-2の素性構造をもつノードSは①の書換えでNP, VPのノードを作る。NPのノードには意味構造が与えられない。VPのSUBCATに入り、書換えが進んで、Vにより詳細な指定がされるまで待ち状態となる。

2) VPノードに「V →VX」が適用され(②)、Xノード(図ではVP3)は、VP2のSUBCATに追加され待ち状態となる。

3) VPノードはVに書き換えられ(③)、さらにターミナル"had"に書き換えられる(④)。"had"には、SUBCATの2番目(VP3)の意味記述と同じということ、VP3が過去分詞であることが記載されている。これでVP3に意味構造(図3-2のTRANS素性そのもの)が与えられたことになり、VP3の書換えが可能になる。

4) X(VP3)が、⑤、⑥と書き換えられ、"slept"まで書き換えられる。ここで最初のNPに意味構造が与えられ、"arthur"への書換え(⑦)が可能になる。

4 文法記述

4.1文法の選択

解析システムと辞書･文法規則を共有させるという目標があるので、文法記述の選択にあたっては、既存のパーサのものを採用することにした。ユニフィケーション文法を採用したパーサとしては、

D-PATR
LFGグラマーライターズワークベンチ
Nadineパーサ

などがある。LFGは純粋なユニフィケーション文法という訳ではなく、文法の記述に自由度が低いために除外した。D-PATRとNadineパーサは、どちらも純粋なユニフィケーション文法で、候補としてふさわしいと言える。

現在は、拡張の可能性を考慮して、Nadineパーサの文法を採用している。英語に対して適用させるため、D-PATRを参考にして以下のような拡張を行っている。

1) テンプレート

D-PATRにもNadineパーサにもテンプレートの機能があり、辞書などで同じような記述を何度も行わねばならない場合に、記述量を減らしたり、誤りを少なくするのに役立っている。D-PATRの場合、テンプレート中で他のテンプレートが使うことができるため、同様の機能をもたせた。

また、英語の場合には同じ語幹でいくつかの語形変化を辞書登録しなければならないため、語幹テンプレートも導入した。語幹テンプレートは、1つのテンプレートで2つ以上のエントリに展開される場合があることを除いて通常のテンプレートと変わらない。

Nadineパーサの場合にはテンプレートは先頭に"!"のついたシンボルで表現されていたが、(テンプレートとして登録してある)シンボルならば何でもよいように変更した。これもD-PATRと同様である。

2) レキシカルルール

英語では、語形変化によって辞書の素性を単に追加するだけではなく、大幅に変更しなければならない場合がある。例えば、受動態では、目的語が主語になる。このため、過去分詞の辞書エントリに対して、現在形(または原形)では目的語に書かれていた制約を主語に対する制約として書かねばならない。このような大幅な変更に対してD-PATRではレキシカルルールという記法を用意しているが、同様の記法を採用した。

レキシカルルールは、inとoutという2つの素性をもつ素性構造で、辞書エントリに対してそこまでに組み上がっている素性構造がin素性とユニファイできたときに、辞書エントリに対する素性構造はout素性におきかえられる。

4.2生成のための拡張

ユニフィケーションベースとした場合、ルールを解析とは逆に適用するのに、適用すべきルールを選ぶのには意味素性を用いなければならない。ところが、単に「書き換え＋制約」で表されているだけでは、適用すべきルールを選ぶのには不十分な場合がある。

次の例を考える。

```
V -> V X                                                    (1)
   <0 TRANS> == <1 TRANS>
   <1 SYNCAT REST FIRST> == <2>
```

これは、複数の書き換え規則をひとつの規則で表そうとしたもので、通常ならば、

```
V -> Aux V     be, have (現在進行形、完了形、受動態などを含む)
V -> V NP      目的語、補語 (2つとる場合は2回適用される)
V -> V Comp    that節など(補語のひとつ)
```

となるところである。また、Vを左辺にもつ規則としては、

```
V -> V P                                                    (2)
   <0 TRANS> == <1 TRANS>
   <1 SYNCAT FIRST> == <2>

V -> terminal symbols
```

などがある。

一方、VのTRANS属性は、例えば、

```
[[RELATION "AT"
 [ARG1     [[RELATION    "PLAY"]
            [ARG1        [ダイエーホークス]]
            [ARG2        [野球]          ]]
 [ARG2     [平和台球場]                ]]
```

となっている。この場合、“AT”のレベルの素性構造に対しては(2)「V →VP」を、“PLAY“のレベルの素性構造に対しては(1)「V →VX(NP)」を適用しなければならない。しかし、“AT”のレベルと“PLAY“のレベルでの構造はほとんど同じで、違いがあるとすれば、それはrelation素性の名前だけであろう。

書き換え規則だけでは、どの規則が適用されるかを決定できない。もしそのまま適用した場合、適用が進んでいってターミナルに達した段階で初めて正しかったかどうかが分かる。また、同じルールを何度も適用してターミナルに達しない場合もある。

これを避けるために、書き換え規則に、親の素性構造の形を(おおまかに)指定する記述を設ける。これをASSERTIONSと呼ぶ。relation素性のように意味情報のうち辞書項目で決まるものには、タイプをあらかじめ与えることておき、ASSERTIONSで指定できるようにする。

このため、書き換え規則「V →VP」には

```
(:ASSERTIONS (:EQ (:TYPE <0 TRANS RELATION>) 'P))
```

と、書き換え規則「V →VX(NP)」には

```
(:ASSERTIONS (:EQ (:TYPE <0 TRANS RELATION>) 'V))
```

と記述しておく。TYPEは辞書項目で決まるもので、規則の読み込みと同時に記録される。現在は、その単語の品詞が与えられているが、さらに細かい分類ができるように、文法記述者に開放する予定である。

現在ASSERTIONSとして記述できる形式は、以下のとおりである。関数適用により容易に機能の増強は可能であるが、現在はこの程度に制限している。

```
:EQ
:AND
:OR
:TYPE
:PATH        (<と>の記法が利用できる)
```

ASSERTIONSの意味は次のように考えることができる。

1) 解析における検証用

ASSERTIONSを追加することは解析にとっては不要な拡張のように見える。しかし、解析時の検証用にも用いることができる。

ユニフィケーションで、ある規則が適用されたとき、意図していない形または場所でユニフィケーションが成功している場合がある。これは、さらに上に組み上がるときに、誤ったユニフィケーションを引き起こす可能性がある。ASSERTIONSがあれば、ある時点における素性構造のあるべき姿を明確に記述できるので、ASSERTIONSによる2重のチェックにより、このような問題を文法から排除することができる。

2) 生成条件のレベル分け

生成時にASSERTIONSでチェックしていることは、(TYPEの指定を除いて)ユニフィケーションを用いて同じようにチェックできるはずである。しかし、ユニフィケーションの素性は全てフラットに並び、どれがチェックに効果があるか(失敗につながりやすいか)を指定できない。チェックに効果があるものを先に行えば、無駄なユニフィケーションが減らせる。ASSERTIONSはチェックに効果がある制約を先に行うことを意味しており、全体の高速化に効果がある。

5 文法規則

英語および日本語に対して、比較的簡単な文法規則、それに対応する辞書を作成した。英語はD-PATR (Karttunen, 1986) ののFEB86DEMOを、日本語はNadineの日本語文法(吉本氏作成)をベースにして、第4章で示したように、ASSERTIONSを加えることによりループを防ぐなどの変更を行っている。

5.1英語文法

5.1.1 扱っている言語現象

ベースとしているD-PATRの文法FEB86DEMOは、以下の文、節の組立てをカバーしている(†のつけられたものは意味表現が正しく抽出できていないなどの不備な点があるもの)。

平叙文、疑問文、命令文
否定文、否定疑問文†
Wh疑問文†
助動詞(進行形、受動態†、完了形を含む)
形容詞、前置詞句による修飾
関係詞節
That節
COMPのコントロール(Equi-NP, Raising)

なお、D-PATRでは、ギャップを比較的簡単に扱う機構が備わっており、この文法で利用されている。これは、あるサブツリーの中でギャップが生じる場合、それに入るもの(フィラー)を文法規則中で指定しておけば、ギャップが生じるところを文法規則中で明示しなくてもよい。ギャップが生じる可能性のある場所はすべて、ギャップになっているかどうか自動的にチェックする機構がパーサに組み込まれている。[1]

ただし、この機構は、フィラーが先行詞として現れる英語には適しているが、フィラーが後に現れる日本語には向いていない。このためNadineパーサには取り入れられていない。同様に、こ

---

1. 例えば、関係代名詞を用いた構文の場合、

```
SREL -> NP S
    <2 gapIn> == <1>
```

によって、NPがSの中のギャップのフィラーとなることを指定する。Sの下で

```
VP -> V NP
```

という規則でNPが存在しなかった場合自動的にこのフィラーが用いられる。この機構がないとすると、この規則とは別に

```
VP -> V
    <0 object> == <0 gapIn>
```

という規則が必要になる。主語がギャップになる場合、間接目的語がギャップになる場合など、それぞれに対してこのような規則を追加する必要がある。

の予備実験用生成システムでも取り入れていない。この問題は、ギャップが生じるところを文法で明示することによって回避できるが、現在は文法では扱っていない。

現在の文法で扱っているのは、

平叙文、命令文
助動詞(現在進行、受動態、現在完了を含む)

のみである。以下の言語現象は、ギャップを含んでいて現在扱っていないものである。

疑問文、Wh疑問文
関係詞節
疑問詞節

また、以下のものは検討未了のものである。

That節
COMPのコントロール(Equi-NP, Raising)

5.1.2 D-PATRからの変更点

文法自体に大きな変更は加えていない。無限の適用を避けるためと、ルールの適用をコントロールするため、ASSERTIONSを以下のところで加えた。

1) ルールの適用のコントロール

先に述べた通り、次の2つのルールで得られる素性構造はほぼ同じ形式をしている。

```
V -> V X
V -> V P
```

適切なルールが選択されるように、意味素性のrelation素性のタイプにより、次のようなASSERTIONSをつけてコントロールする。

```
V -> V X
   (:ASSERTIONS (:EQ (:TYPE <0 trans relation>) 'V))

V -> V P
   (:ASSERTIONS (:EQ (:TYPE <0 trans relation>) 'P))
```

2) SYNCAT(SUBCAT)の数の制限

```
V -> V X
   <1 syncat first> == <2>
   <0 syncat> == <1 syncat rest>
```

というルールが解析で適用されると、動詞の辞書から得られたsyncatのリストが1つ1つXによって埋められて、リストの長さが1つ減る。逆に生成で適用されると、リスト長さが1つ増えることになる。ターミナルまでいかないと正しい長さは分からないので、無限に適用され

ることになってしまう。このため、このルールには、syncatの数を制限するASSERTIONSを追加した。

現在は全ての動詞でsyncatの数を一様に制限している。しかし、この数は動詞のタイプによって決定されるものなので、それを用いてコントロールすることが考えられる。現在のところ動詞をタイプ分けしていないが、このようなコントロールは必要である。

5.2 日本語文法

日本語の文法は、この生成システムが日本語に対しても適用可能か否かを調べるために作成したものである。このため、扱っている文は1文だけであり、言語現象のどこまでをカバーしているかなどはこの段階では言えるものではない。

5.2.1 キーボード会話解析用文法からの変更点

日本語の文法は、吉本氏のキーボード会話解析用の文法(吉本&小暮, 1988)を利用している。生成に利用するために以下のような変更を加えた。この文法では、補助動詞、助動詞の語順まで詳細に指定されているため、この程度の変更で正しい文が生成できるようになった。

1) 修飾語の排除

キーボード会話解析用の文法による素性構造では、修飾語は、それぞれの修飾のタイプによる素性を与えられ、意味素性のrelationおよび格マーカ素性と同じレベルにはいる。例えば、「登録用紙は既にお持ちでしょうか?」における副詞「既に」は、

```
[[reln  持つ-2]
 [agen  [*hearer*]]
 [obje  [登録用紙]]
 [tloc  [既に]]]
```

のように、「持つ」のargumentに入るagen, objeなどと同様のtloc(TIME-LOCATION)格に入る。このようにした場合に、解析時には、規則

```
V -> ADV V
   <2> == <1 head coh>

ADV -> 既に
   <1 head coh sem> == [[tloc ...]]
```

によって、左辺のVへこのtloc格を与えることができるので、これでもよい。しかし、ユニフィケーションベースの生成では、この格をもつ左辺のVがあったときに右辺のVにもこの格は残ってしまい、無限に同じ規則が適用されることになってしまう。

現在の文法では、この問題のため、副詞を扱っていない。

解決策のひとつとしては、英語と同様に、修飾は文全体にかかる、すなわち、ひとつの修飾ごとに全体の意味素性をつつみこむような解釈をとることが考えられる。先の表現では、例えば、

```
[[reln  tloc]
 [arg1  [[reln  持つ]
         [agen  [*hearer*]]
         [obje  [登録用紙]]]
 [arg2  [既に]]]
```

とする(arg1, arg2よりも適切な素性名が必要である)。

この方法を採用すると、副詞の語順と意味表現が1対1に決まってしまい、副詞の語順のゆれを意味表現で吸収できなくなってしまう。生成は容易になるが、変換部は副詞の語順まで考慮しなければならず、余計な負担がかかることになる。また、これは、変換部と生成部の仕事の分担の問題にもなる。

もう一つ考えられる解決策は、先の

```
V -> ADV V
```

規則において、ADVによって占められた格が右辺のVには渡されないような規則を書くことである。これを規則だけで吸収するには、例えば各素性ごとに、

```
V -> ADV V
   <0 sem tloc> == <1 sem>
   <2 sem tloc> == -
   <0 sem ploc> == <2 sem ploc>
      :
      :
   <0 sem reln> == <2 sem reln>
   <0 sem agen> == <2 sem agen>
      :
```

のような、自分の素性以外をそのままパスするような規則を作らねばならない。

規則だけで吸収することに拘泥しなければ、例えばLFGのように、素性名にも変数が使えるような機構にし、対応するもの以外はそのままパスできるようにすることが考えられる。

2) SUBCATの数の制限とSLASH

日本語においても英語の場合と同様に、格要素を動詞と対応づけるためそれぞれの動詞にSUBCATというリストが保持されている。日本語の場合は格要素の省略も可能であるため、これ以外に省略された格要素のリストを保持するSLASHというリストも存在している。SUBCATを埋める格要素がなくなった時点でSUBCATからSLASHへ要素が移される。

このルールは、効率を落とす原因になっていた。また、英語の場合に与えていたSUBCATの長さの制限を、SLASHへも同様に与えなければならない。このため、今回はSLASHを取り入れず、SUBCATのみを用いている。副助詞「は」は、SLASHにはいっているものに対して働くようになっていたが、SUBCATにはいっているものに対して働くよう変更した。

6 生成結果

6.1 英文生成結果

付録1の入力素性構造に対しての、英文生成結果を付録2に示す。付録1の入力素性構造は、想定される英文を解析して得られた素性構造から、意味素性(TRANS素性)と文の形態を指定するいくつかの素性(HEAD素性の一部)を抜きだして作ったものである。

6.2 日本語文生成結果

付録3に日本語文の入力を、付録4にそれからの生成結果を示す。入力は英文のときと同様に解析結果から作った。この2つの例は、TOPIC素性の値を変えてみたものである。最初の例は「登録用紙」をトピックにしたもの、次の例はトピックはないと指定したものである(TOPIC素性の値を"-"にしたもの)。

6.3 問題点と解決策

目的とする文は一応得られているが、以下のような問題点がある。

1) 生成にかなりの時間を要している。
   現在のルールの適用のコントロールには無駄な部分が多い。この問題については7章で検討する。

2) 誤った文が生成される。
   文法が不備であるために生じている問題である。英語の文法(FEB86DEMO)の場合、名詞句に対する素性構造が統一されていないために起こる。

(2)に対する対策として、素性構造のパターンとターミナルとの関係を導入して、ターミナルへの書き換えはパターンを満たしていなければならないという制限をつけることが考えられる。このようなターミナルへの書き換えの制限に関しては、7.1.3で検討する。ここでは、素性構造を統一するように文法を変更することを試みた。また、メカニズムを変更しない範囲(文法の修正ですむ範囲)での改良を試みた。その結果を6.4に示す。

6.4 文法の改良

文法を変更して、それが生成結果(特に生成に要する時間)にどのような影響を与えているかを調べた。

1) 名詞に対する素性構造の統一

最初の文法での普通名詞の形式は、

```
[[RESTRICTION   [[RELATION    (辞書エントリで決定される部分)]
                 [ARG1        ?X01]]]
 [PARAMETER     ?X01]]
```

で表される。一方、固有名詞は、

```
[[CONDITION  [[RESTRICTION   [[RELATION NAMED]
                              [ARG1  ?X01]
                              [ARG2  (辞書エントリで決定される部分)]]]
               [PARAMETER    ?X01]]
 [INDIVIDUAL NIL]]
```

で表現される。これらはユニフィケーション可能である。“that”等の指示代名詞は普通名詞と同じ形式であるため、固有名詞である“uther”とユニファイされ、「正しい書き換え」とみなされていた。

意味表現の意味論的に問題があるかもしれないが、素性構造をすべて固有名詞のフォーマットに合わせて生成実験を行った。この入力素性構造を付録5、その生成結果を付録6に示す。

無駄な生成を行わなくなったが、時間的にはほとんど変わっていない。これは、名詞句に対してにTRANS素性が付与されるのが、動詞がすべて決定した後になるため、ここで複数の枝わかれが起こっても他に影響を与えないからである。

なお、FS-5とFS-6の生成はこのバージョンからテストしている。最初のバージョンでは、主語、目的語、第2目的語のそれぞれで複数の生成を行い、それが文としては組合せ的爆発を生むからである。

2) ノンターミナルとプレターミナルの分離

現在のメカニズムでは、ターミナルへの書換えとノンターミナルの書き換えの区別を行っていない。例えば、動詞句と動詞が同じシンボルVで表されるため、動詞句に対してすべての動詞ターミナルとの書換えを試している。これは、メカニズムの改良で解決されるべきであるが、ここでは動詞句と動詞のシンボルを変えて実験を行った。この結果を付録7に示す。今までの結果に対して約1/2の時間で生成ができるようになった。これより、ターミナルへの書換えとノンターミナルの書き換えを区別することの重要性がわかる。

3) それ自身に意味素性が与えられていない助動詞の扱い

助動詞は、通常それ自身に対する意味素性が辞書記述に与えられている。例えば、“may”の場合の意味素性は、

```
<0 TRANS RELATION> == "MAY"
<0 TRANS ARG1> == <0 SYNCAT REST FIRST TRANS RESTRICTION>
```

で与えられる。これは、“may”がない場合の文全体のTRANS素性(これは「V→VX」のXのほうから与えられる)を、RELATION素性“may”のARG1にして1段埋めこんだ形になる。

一方、“have”の場合は意味素性を与えられておらず、その代わりに、Xの位置に来る動詞が意味素性を与えること、Xの位置に来る動詞が過去分詞であることだけが指定されている。過去分詞の辞書記述で意味素性は、

```
[[RELATION    PERFECTIVE]
 [ARG1        [[RELATION  (動詞の意味)]
               [ARG1       ... ]]]]
```

という形式で与えられるため、これにより、文の意味にPERFECTIVEが反映されることになる。

この文法を生成に用いた場合、ターミナルへの書換えの際にPERFECTIVEでないものも“have”に書き換え可能になる。PERFECTIVEでないものは、PERFECTIVEでないにもかかわらずXの位置に来る動詞が過去分詞であるという指定がされているため、これに対応するターミナルがないことにより失敗する。進行形や受動態を表すbe動詞も同様に失敗する。これらの失敗は木構造のコピーを作って何度も書換えを行ったのちに判明するので、影響が大きい。

これを避けるために“have”にRELATION＝PERFECTIVEだけを指定するTRANS素性を与えた。また、進行形を表すbe動詞にも同じようにPROGRESSIVEであることの指定を与えた。受動態ではそのようなRELATIONが特別に表される訳でく、単にARG1が無指定になるだけなので、受動態を表すbe動詞に対しての制限はこの事実を用いた。これによる生成結果を付録8に示す。劇的な効果が得られていることが分かる。なお、解析ではこの記述は冗長であるだけで害はない。

この中でFS-5(受動態の文)のみは、他のものほど大きな効果は得られていない(それでも1/2程度になってはいるが)。これは、be動詞が“am”, “are”, “is”の3つに枝わかれし、それぞれ書換えが進んでいって構文上の主語の位置にはいる“he”が来た時点で、適切な“is”以外が落ちたためであると考えられる。ユニフィケーションに基づいて辞書項目に多くの情報をもたせるシステムでは、動詞と主語の両方の書換えがすんだ時点で初めてアグリーメントをとることができる(もちろん、文法のモジュラリティを犠牲にしてルールで対応することは可能であるが)。この問題に関しては7.1.4で扱う。

4) 適切な規則の選択

現在のバージョンでは、動詞のタイプによって書き換え規則の選択していないことは、5.1.2(2)に記述したとおりである。すなわち、全ての動詞は、最大3個のSUBCATフレームを持っているとみなされ、書換えが進められる。

動詞をタイプ分けして生成実験を行った結果を付録9に示す。ここでは、動詞をMonadic, Dyadic, Triadicの3種類に分けた。それぞれ、SUBCATを1個、2個、3個とるものである。これに合わせて、「V→VX」の規則を次の3つに分けた。

```
V -> Monadic
   (:assertions
      (:eq (:type <0 trans relation>) 'Monadic))
```

```
V -> Dyadic X
   <1 syncat rest first> == <2>
   (:assertions
      (:eq (:type <0 trans relation>) 'Dyadic))

V -> Triadic X X
   <1 syncat rest first> == <2>
   <1 syncat rest rest first> == <3>
   (:assertions
      (:eq (:type <0 trans relation>) 'Triadic))
```

なお、受動態の場合には、レキシカルルールにより、カテゴリTriadicはDyadicに、DyadicはMonadicに変えられる。一方、RELATION素性のタイプは変わらないので、例えば2番目の書き換え規則のASSERTIONSには、受動態でタイプがTriadicという条件も含めなければならない。

また、PROGRESSIVEおよびPERFECTIVEの場合もこの規則が選択されるようにする。ところが、この場合に2番目の規則の右辺のXはVになるのだが、動詞"be"または"have"により、このTRANS素性には左辺のVと全く同じものが与えられるのである。ここで違ってくるのはHEAD素性のうちのFORM素性がFINITEからPASTPRT(過去分詞)になったところだけである。これにしても、完了進行形などの形式を考えれば、例えば進行形として第2の規則が選択されるのはFINITEのときだけではなくなるのである。付録9ではこのような選択を行うような拡張を怠っており、このためFS-2は生成に失敗している。

まだ正しい文法にはなっていないが、適切な選択があれば生成時間を大幅に減らせることが分かったことで、この改良は一応終了させる。最初の文法で40分かかっていたものが、メカニズムの変更なしで、とにかく2秒ですむようになった(表6.1参照。FS-1の場合)。一方、このために文法が複雑になったことも事実である。第7章ではこれらの結果をもとに、今後の課題を検討する。

	文法1	文法2	文法3	文法4	文法5
FS-1	2357.4	2395.8	1233.8	67.2	1.9
FS-2	2722.3	2698.0	1246.9	68.0	-
FS-3	1954.1	1620.0	691.4	54.6	13.7
FS-4	6401.78	6372.9	3172.0	273.3	13.3
FS-5	-	5407.6	2556.9	1118.8	120.6
FS-6	-	5322.0	2647.8	196.0	20.9

表6.1 生成時間一覧(単位:秒)

7 問題点と解決の方向

現在の生成システムは予備実験用として作成したもので、問題点を数多く含んでいる。この中には既にふれた部分もあるが、それらを含めて問題点を整理する。

7.1速度の向上

まず課題としてあげられるのは、効率の向上である。このためには、以下のようなことが考えられる。

7.1.1 同じルールの繰り返し適用の排除

ルール間で素性構造を伝搬させており、共有させているため、あるルールを適用した結果が局所化されず、句構造のどこまで影響を及ぼすかが分からない。このため、ある木構造のノードに対して適用した結果が2つ以上ある場合は、木全体をコピーしている。全く同じ素性構造をもつノードが生成に失敗したならば、あるいは逆に生成に成功したならば、同じルールの適用は行わなくて良いはずである。現在のところそのような成功/失敗の情報を用いておらず、何度も同じ規則を適用するようになっている。

これを避けるために、2つの方法が考えられる。

1) 履歴の管理
書き換え規則の適用その結果を蓄積し、書き換え規則適用の際に素性構造の同一性をチェックする。失敗につながる素性構造は、それ以降の適用をやめる。成功につながる素性構造は結果を利用する(ただし結果が局所化されている場合に限る)。同一性のチェックにはかなりの時間がかかることが予想されるので、毎回行っていたのでは速度の向上につながらない恐れがあるが、ルールの複数回の適用をまとめて保持することにより効率化が期待できる。

2) Structure Sharing
Nadineパーサ(ユニフィケーション機構)の新バージョンでは、選言(disjunction)がシステムで管理される。選言で結ばれた2つの素性構造は共通部分と固有部分にわけられ、自動的にメインテナンスされる。この機構を利用するように生成システムを変更することで、この問題は自然に回避される。すなわち、適用の結果が失敗であっても成功であっても(局所化されない影響を含めて)、適用の結果は利用されることになる。

7.1.2 適用するルールの選択

適用される書き換え規則の選択は、カテゴリとASSERTIONSで行っている。ASSERTIONSが適切に付加されていないと、無駄な規則を適用していき、ターミナル(辞書エントリ)まで達して初めて失敗することが起こる。

ASSERTIONSを強化するためには、ASSERTIONSの記述能力を強化することも必要である。例えば、TYPEでは品詞がそのまま用いられているが、品詞をさらにタイプ分けしたい場合がある。SUBCATのリストが無限に長くなるのを防ぐ規則では、動詞を細分化し、それによってSUBCATのリストの長さの最大値を変えることができる1。

ASSERTIONSを強化する以外の対策として、ASSERTIONSに対応するものをシステムが自動作成することが考えられる。もともとこれは、ある時点での素性がどうなっていなければならないかを記述したもので、ルールと辞書の集合が与えられれば、そこから計算できる可能性がある。現在の生成システムが完全にトップダウンに行われるのに対して、ルールと辞書の集合を用いてボトムアップ予測を加えるものと考える。しかし、6.5で示したようにルールの適用の条件はTRANS属性に限る訳ではないので、この方法には困難が予想される(原理的に解決不可能な問題かもしれない)。

7.1.3 ターミナルへの書き換え(1)

ターミナルへの書き換えは、適用するルールの選択の特殊な場合である。現在のシステムではターミナルへの書き換えを特別扱いしておらず、また、文法がプレターミナルシンボルとノンターミナルシンボルを区別していないため、どのような場所においてもターミナルへの書き換えを試しており、速度を遅くする大きな原因になっている。また、ターミナルへの書き換えに際しても、(同じカテゴリの)全てのターミナルへの書き換えを試している。

これを避けるために、単語に特有の素性値をもつ素性の位置(パス)を記憶しておく。例えば、現在のFEB86DEMO文法で

普通名詞	<0 TRANS RESTRICTION RELATION>
固有名詞	<0 TRANS CONDITION RESTRICTION ARG2>
動詞	<0 TRANS RELATION>
動詞進行形	<0 TRANS ARG1 RELATION>

などである。ルール読み込み時にこのパスとターミナルとの関係を記憶しておく。プレターミナルシンボルとノンターミナルシンボルを区別しておくことを前提にしておき、プレターミナルが来たときには、このパスの値をキーとして書き換え規則をアクセスする。

7.1.4 ターミナルへの書き換え(2)

6.3 (3)で検討した問題に、アグリーメントの問題があった。例えば、be動詞現在形の“am”, “are”, “is”の3つのうちから適切なものを選択するのには主語の情報が必要であるのに対して、主語が決定されるためには動詞が決定されてSUBCAT素性から必要な情報を得なければならない。このため、結局すべてのターミナルに書き換えてみて、複数の木のコピーがそれぞれ展開されて、最終的に一致がとれないものが落ちるのを待つということになる。

しかし、“am”, “are”, “is”の3つで異なるのは、人称と数だけで、その他は全く同じである。それならば、この書換えを次のように2段階に行うことが考えられる(さらに多段階書換えにすることも考えられる)。

---

1. 6.4 (4)では動詞を細分化したものをそのまま品詞に用いただけで、ここでの細分化とは意味が異なる。ここでの細分化は、品詞を階層化し、例えば動詞でありかつMonadic動詞であるような単語を考えることである。

V → be-present → am
are
is

または、7.1.1であげたStructure Sharingをここにも反映させることでも、解決策のひとつとして考えられる。

2段階書換えの場合には、be-presentのところで主語が決定されるまで書換えをペンディングにしなければならない。現在のところ、ペンディングの条件は、「意味素性が与えられていない場合」となっているが、この条件を、それぞれのルールに応じて変更できるようにしなければならない。

7.2生成機構の拡張

7.1で述べたのは高速化のための改良点であるが、生成機構の機能面での充実も望まれる。ここでは、拡張すべき点を考える。

7.2.1 優先度

複数の生成結果が出されることは、文法の作成時には有用であるが、実際のキーボード会話や電話会話では1つの結果が得られれば(もしそれが正しいものならば)十分である。また、1つの結果だけを早く出すようにもできる。

この際には、よい生成結果をなるべく早く出したいので、何らかの優先機構を入れることが考えられる。ルールと書換えの対象となる素性構造に対して優先度をつけておき、書換えの際にこれらから総合の優先度を計算し、最も優先度の高い素性構造とルールの対から順に書換えを行っていくようにする。現在のペンディングツリーリストは、この拡張のための出発点にもなりうる。

7.2.2 文法開発ツール

先にのべたような無限に適用される規則ができてしまうのは、現在の機構では避けられない。しかし、少なくともその規則が無限に適用されることは避けたい。現在のところ、トレースだけで開発ツールと呼べるものは何もないが、開発ツールの整備は重要である。

- ○ トレースの強化
  (ブレークポイント、繰り返して適用されていることの警告など)
- ○ 木構造の表示＋そのノードに対する素性構造のインスペクト

7.3文法の強化

先に述べたとおり、現在の文法は言語現象の一部しかカバーしていない。ここでは、どのように拡張してくべきかを述べる。

7.3.1 ギャップの扱い

ギャップをどのように扱うかには2通りが考えられる。

○ ギャップを扱うように、メカニズムのほうを変更する
解析機構も同時に変更する必要がある。

○ ルールで対処する
ルールの数はかなり増えるが、ギャップが生じるところを明示的に規則に表現できるのが利点である。

先に述べたように、D-PATRにおいてギャップを扱う機構は、英語の文法にかなり依存している。このため、ルールで対処するという方向を採用する。

このためのルールの例としては、ギャップが生じているところで、

```
V -> V X
   <1 syncat rest first> == <2>          ;XがVのSUBCATの第2要素に入る
   <0 syncat rest> == <1 syncat rest rest>
```

に対応するものとして、

```
V -> V
   <1 syncat rest first> == <0 gapIn>
                                         ;ギャップがVのSUBCATの第2要素に入る
   <0 syncat rest> == <1 syncat rest rest>
```

を追加する。ギャップが生じる可能性のあるところには、同様のルールを追加していかなければならない。また、ギャップが生じないところでは、ギャップ(gapIn素性)を適切に伝搬させなければならない。

7.3.2 入力素性構造の検討

素性構造の選択は、文法規則の既述に影響を与える。5.2で問題にしたように、修飾語が格マーカ素性と同じレベルに入ると、生成では規則に工夫が必要になる。生成機構の変更も考える必要がある。

修飾に関する素性構造の表現形式に関してまとめると、次のようになる。

1) 格マーカ素性と同じレベルに並べる。

   修飾語の語順のゆれを素性構造で吸収できる。
   生成規則が複雑になる。

2) ひとつの修飾ごとに全体の意味素性をつつみこむ。

   変換部で修飾語の語順まで考慮しなければならない。
   同じタイプの修飾を2個以上もたせることができる。

意味素性の意味論を考慮する必要がある。スコープを表すと考えると、この2つが混在させなければならないことも考えられる。

素性構造の決定には、同じレベルの素性構造を出力する日本語構文解析、それを入力とする日本語意味解釈機構、さらに、変換機構も関係してくる。

7.3.3 イディオムの扱い

イディオムの扱いは、辞書に記述する方法と、規則として記述する方法がある。LFG (Kaplan & Bresnan, 1982)では、辞書に記述する方法を採用しており、"observe"という意味の"keep tabs on"では、

```
keep:  V, (↑ PRED) = 'OBSERVE<(↑ SUBJ) (↑ ON OBJ)>'
          (↑ OBJ FORM) =c TABS
```

というように、動詞"keep"の場所に(通常のkeepの意味とは別に)記述される。現在の英語文法の記述形式では、

```
<0 trans relation> == observe
<0 trans arg2> == <0 syncat rest first trans restriction arg1>
<0 trans arg1> == <0 syncat first trans>
<0 syncat rest rest> == end
<0 syncat rest first cat> == P
<0 syncat rest first head lex> == ON
```

となる。ルールで記述する場合、

```
V -> V P
   <1 head stem> == keep
   <2 head lex> == ON
   <0 trans relation> == observe
   <0 trans arg2> == <0 trans restriction arg1>
   <0 syncat first> == <1 syncat first>
```

となる。イディオムのためのルールが通常のルールと混在し、イディオムのためのルールが散逸するので、メインテナンスも困難になる。このため、辞書に記述する方法をとる方が望ましい。

8 今後の課題

以上、現在の予備実験用生成システムの内容、および問題点について報告してきた。生成システムに対して、ユニフィケーション機構を適用する可能性は確かめられた。

しかし、この機構は、設計の章で述べたように、翻訳システム/通訳システムの生成部としては十分ではない。第8章で述べた短期的な課題だけではなく、翻訳システム/通訳システムへ発展させるための課題が残されている。

ここでは、そのような長期的な課題を列挙する。これらは、十分な検討がなされているものではない。

8.1英語における発話意図の研究

発話意図(Intention)を発話内容(Proposition)と分離するということが、キーボード会話翻訳システム、自動翻訳電話の自然言語処理の研究におけるテーマである。キーボード会話翻訳システムの日本語解析部では、発話意図も解析するように文法規則が作られており、この面での研究は進んでいる。

一方、英文の生成の研究は、文法に関しては既存のものを利用している状態である。言語学的な研究はまだ手がつけられていない。

発話とその発話意図との関係の整理から始めて、解析および生成が実験で確かめることができる形式の文法の開発へ繋げていかねばならない。

8.2How to sayのプランニング

この予備実験用生成システムの設計方針として、表層のスピーチアクトを入力とすることをあげた(方針1)。これはテキストプランニングを行わないことを意味していた。

自然言語インターフェースなどにおける生成においては、

1) Planning: "What to say"を決定する
2) Realization: "How to say"を決定する

の2つのステージがあると言われている。この場合、翻訳においては、(スピーチアクトも含めて)"What to say"は既に決定されているので、Realizationだけを行えばよいということになる。

しかし、Kogure (1988)が示したように、深層のスピーチアクトも考慮した場合には、考慮すべきことが変わってくる。深層のスピーチアクトと表層のスピーチアクトは1対1対応しないと考え

られる。また、この対応は日本語と英語では異なることも考えられ、これらの対応の研究を始める必要がある。

また、例えば、「分かりやすい話しかた」ということを考えた場合、日本語と英語では、話の構成を含めて異なってくることが考えられる。この場合は、生成において文章の構成からやりなおす必要が出てくる。こうなると始めにテキストプランニングを行わないという前提が崩れ、"How to say"をプランニングしているともいえるだろう。翻訳システムとして考えた場合は、ここまで行う必要はないとも言えるが、通訳システムの場合は考慮に値すると思われる。

8.3音声合成とのインターフェース

通訳システムの生成として考えなければならなことの1つに、音声合成とのインターフェースがある。発音はほぼ一定であるとしても、イントネーションは、文中の位置や、発話意図、トピックなどによって変わってくると考えられる。このような情報を音声合成に適切に渡すためには、生成部が適切に付加しておかねばならない。

イントネーションと発話意図やトピックなどの関係は、現在まだ解明されていないということだが、今後このような研究も行っていかねばならないだろう。

謝辞

この研究の機会を与えてくださったATR自動翻訳電話研究所樽松社長に感謝いたします。また、この研究を進める際にアドバイスしてくださいました言語処理研究室相沢室長、小暮氏、吉本氏、Mr. Emele、その他言語処理研究室の研究員諸氏に感謝いたします。

参考文献

Appelt, D. E. 1987: "Bidirectional Grammars and the Design of Natural Language Generation", TINLAP-3.

Kaplan, R. M. and Bresnan, J. 1982: "Lexical-functional grammar: A formal system for grammatical representation", In J. Bresnan (ed.), "The mental representation of grammatical relations", Cambridge, MIT Press.

Karttunen, L. 1986: "D-PATR: A Development Environment for Unification-Based Grammars", Report No. CSLI-86-61, Center for the Study of Language and Information, Stanford.

Kogure, K. 1988: "A Method of Analyzing Japanese Speech Act Types (I) -- Combining Unification-Based Syntactico-Semantic Analysis and Plan Recognition Inference", ATR Technical Report TR-I-0026.

McKeown, K. R. 1985: "Discourse Strategies for Generating Natural Language Text", Artificial Intelligence 27, 1-42.

吉本、小暮 1988: "句構造文法にもとづく日本語の解析", ATR Technical Report TR-I-0049.

```
;;; -*- Syntax: Common-lisp; Default-character-style: (:FIX :ROMAN :NORMAL); Mode: LISP -*-

(setq fd-1 '((<0> == [[HEAD  [[FORM  FINITE]
                              [TENSE  PAST]
                              [mode declarative]]]
                      [TRANS  [[RELATION "sleep"]
                               [ARG1 [[CONDITION [[PARAMETER  ?X11]
                                                  [RESTRICTION [[ARG1  ?X11]
                                                                [RELATION  NAMED]
                                                                [ARG2  "uther"]]]]]
                                      [INDIVIDUAL  NIL]]]]]]])))


(setq fs-1 (car (make-prod-fss fd-1)))

(setq fd-2 '((<0> == [[HEAD [[FORM FINITE]
                             [TENSE PAST]
                             [mode declarative]]]
                      [TRANS [[relation perfective]
                              [arg1 [[RELATION "sleep"]
                                     [ARG1 [[CONDITION [[PARAMETER ?X11]
                                                        [RESTRICTION [[ARG1 ?X11]
                                                                      [RELATION NAMED]
                                                                      [ARG2 "uther"]]]]]
                                            [INDIVIDUAL  NIL]]]]]]]])))

(setq fs-2 (car (make-prod-fss fd-2)))

(setq fd-3
      '((<0> == [[HEAD [[FORM  NONFINITE]
                        [MODE  IMPERATIVE] ]]
                 [CAT  SENTENCE]
                 [TRANS [[ARG2 ?x09[[CONDITION [[RESTRICTION [[RELATION HEARER]
                                                              [ARG1  ?X13]]]
                                                [PARAMETER  ?X13]]]
                                       [INDIVIDUAL   NIL]]]
                         [ARG3 [[ARG2 [[RESTRICTION [[ARG1 [[CONDITION [[RESTRICTION [[ARG1 ?X21]
                                                                                      [RELATION "
sword"]]]
                                                                                      [PARAMETER  ?X21]]]
                                                                       [INDIVIDUAL NIL]]]
                                                     [RELATION "a"]
                                                     [ARG2  ?X25]]]
                                           [PARAMETER  ?X25]]]
                                  [RELATION  "use"]
                                  [ARG1  ?X09]]]
                         [ARG1 [[CONDITION [[RESTRICTION [[RELATION SPEAKER]
                                                          [ARG1 ?X30]]]
                                            [PARAMETER ?X30]]]
                                [INDIVIDUAL NIL]]]
                         [RELATION IMPERATIVE]]]] )))

(setq fs-3 (car (make-prod-fss fd-3)))

(setq fd-4
      '((<0> == [[HEAD [[FORM FINITE]
                        [tense present]
                        [MODE DECLARATIVE]]]
                 [CAT SENTENCE]
                 [TRANS [[ARG1 [[ARG1 [[CONDITION [[RESTRICTION [[RELATION  MALE]
                                                                 [ARG1 ?X15]]]
                                                   [PARAMETER  ?X15]]]
                                          [INDIVIDUAL NIL]]]
                                 [RELATION "sleep"]]]
                         [RELATION "may"]]]])))

(setq fs-4 (car (make-prod-fss fd-4)))
```

付録1 英文生成入力1

```
Command: (time (gen2 fs-1))

Evaluation of (GEN2 FS-1) took 2357.407590 seconds of elapsed time
including 786.535 seconds waiting for the disk for 15,922 faults.
The garbage collector has flipped, so consing was not measured.
("that slept" "there slept" "who slept" "who slept" "whom slept" "what slept" "uther slept")

Command: (time (gen2 fs-2))

Evaluation of (GEN2 FS-2) took 2722.283208 seconds of elapsed time
including 921.667 seconds waiting for the disk for 18,600 faults.
The garbage collector has flipped, so consing was not measured.
("that had slept" "there had slept" "who had slept" "who had slept" "whom had slept" "what had sl
ept"
 "uther had slept")

Command: (time (gen2 fs-3))

Evaluation of (GEN2 FS-3) took 1954.058796 seconds of elapsed time
including 671.020 seconds waiting for the disk for 14,016 faults.
The garbage collector has flipped, so consing was not measured.
("use a sword" "use that" "use this" "use there" "use it" "use him" "use her" "use them" "use you
" "use me" "use us"
 "use uther" "use arthur" "use gwen" "use cornwall")

Command: (time (gen2 fs-4))

Evaluation of (GEN2 FS-4) took 6401.728420 seconds of elapsed time
including 2068.922 seconds waiting for the disk for 45,227 faults.
The garbage collector has flipped, so consing was not measured.
("that may sleep" "there may sleep" "he may sleep" "who may sleep" "who may sleep" "whom may slee
p" "what may sleep")
```

付録2 英文生成結果1

```
;;; -*- Mode: LISP; Syntax: Common-lisp -*-

(setq fd-1                                          ; a-2-1
      '((<0> == [[HEAD  [[MODL  [[SFP-1  KA]]]
                        [TOPIC  ?TOPIC[[RESTR  [[RELN  登録用紙-1]
                                               [OBJE  ?X28]]]
                                      [PARM  ?X28]]]
                        [cform senf]]]
                [SUBCAT {}]
                [SLASH {}]
                [SEM  [[RELN  S-REQUEST]
                       [AGEN  ?X11[[LABEL  *SPEAKER*]]]
                       [RECP  ?X13[[LABEL  *HEARER*]]]
                       [OBJE  [[RELN  INFORMIF]
                               [AGEN  ?X13]
                               [RECP  ?X11]
                               [OBJE  [[RELN  よう-GUESS]
                                       [EXPR  [[LABEL  *SPEAKER*]]]
                                       [OBJE  [[RELN  持つ-2]
                                               [AGEN  [[LABEL  *HEARER*]]]
                                               [OBJE  ?TOPIC]
                                               [TLOC  [[RESTR  [[RELN  既に-1]
                                                               [OBJE  ?X32]]]
                                                      [PARM  ?X32]]]]]]]]]
                [CAT V]])))

(setq fs-1 (car (make-prod-fss fd-1)))

(setq fd-2                                          ; a-2-1
      '((<0> == [[HEAD  [[MODL  [[SFP-1  KA]]]
                        [TOPIC - ]
                        [cform senf]]]
                [SUBCAT {}]
                [SLASH {}]
                [SEM  [[RELN  S-REQUEST]
                       [AGEN  ?X11[[LABEL  *SPEAKER*]]]
                       [RECP  ?X13[[LABEL  *HEARER*]]]
                       [OBJE  [[RELN  INFORMIF]
                               [AGEN  ?X13]
                               [RECP  ?X11]
                               [OBJE  [[RELN  よう-GUESS]
                                       [EXPR  [[LABEL  *SPEAKER*]]]
                                       [OBJE  [[RELN  持つ-2]
                                               [AGEN  [[LABEL  *HEARER*]]]
                                               [OBJE [[RESTR  [[RELN  登録用紙-1]
                                                              [OBJE  ?X28]]]
                                                      [PARM  ?X28]] ]
                                               [TLOC  [[RESTR  [[RELN  既に-1]
                                                               [OBJE  ?X32]]]
                                                      [PARM  ?X32]]]]]]]]]
                [CAT V]])))

(setq fs-2 (car (make-prod-fss fd-2)))
```

付録3　日本語文生成入力

```
Command: (pprint-fs fs-1)
 ?X01[[0  ?X02[[HEAD  ?X03[[CFORM  ?X04 SENF]
                          [TOPIC  ?X05[[PARM  ?X06[]]
                                       [RESTR  ?X07[[RELN  ?X08 登録用紙-1]
                                                    [OBJE  ?X06]]]]]
                          [MODL  ?X09[[SFP-1  ?X10 KA]]]]]
               [SUBCAT {}]
               [SLASH {}]
               [SEM  ?X11[[RELN  ?X12 S-REQUEST]
                         [AGEN  ?X13[[LABEL  ?X14 *SPEAKER*]]]
                         [RECP  ?X15[[LABEL  ?X16 *HEARER*]]]
                         [OBJE  ?X17[[RELN  ?X18 INFORMIF]
                                     [AGEN  ?X15]
                                     [RECP  ?X13]
                                     [OBJE  ?X19[[RELN  ?X20 よう-GUESS]
                                                 [EXPR  ?X21[[LABEL  ?X22 *SPEAKER*]]]
                                                 [OBJE  ?X23[[RELN  ?X24 持つ-2]
                                                             [AGEN  ?X25[[LABEL  ?X26 *HEARER*]]
]
                                                             [OBJE  ?X05]
                                                             [TLOC  ?X27[[PARM  ?X28[]]
                                                                         [RESTR  ?X29[[RELN  ?X3
0 既に-1]
                                                                                     [OBJE  ?X2
8]]]]]]]]]]]]]
               [CAT  ?X31 V]]]]
NIL

Command: (time (gen2 fs-1))
Evaluation of (GEN2 FS-1) took 585.797344 seconds of elapsed time
including 145.342 seconds waiting for the disk for 2,903 faults.
The garbage collector has flipped, so consing was not measured.
("登録用紙 は お持ち で しょ う か" "登録用紙 は お持ち で しょ う か" "登録用紙 を お持ち で しょ う か" "登録用紙 を お持ち で しょ う
か")

Command: (pprint-fs fs-2)
 ?X01[[0  ?X02[[HEAD  ?X03[[CFORM  ?X04 SENF]
                          [TOPIC  ?X05 -]
                          [MODL  ?X06[[SFP-1  ?X07 KA]]]]]
               [SUBCAT {}]
               [SLASH {}]
               [SEM  ?X08[[RELN  ?X09 S-REQUEST]
                         [AGEN  ?X10[[LABEL  ?X11 *SPEAKER*]]]
                         [RECP  ?X12[[LABEL  ?X13 *HEARER*]]]
                         [OBJE  ?X14[[RELN  ?X15 INFORMIF]
                                     [AGEN  ?X12]
                                     [RECP  ?X10]
                                     [OBJE  ?X16[[RELN  ?X17 よう-GUESS]
                                                 [EXPR  ?X18[[LABEL  ?X19 *SPEAKER*]]]
                                                 [OBJE  ?X20[[RELN  ?X21 持つ-2]
                                                             [AGEN  ?X22[[LABEL  ?X23 *HEARER*]]
]
                                                             [OBJE  ?X24[[PARM  ?X25[]]
                                                                         [RESTR  ?X26[[RELN  ?X2
7 登録用紙-1]
                                                                                     [OBJE  ?X2
5]]]]]
                                                             [TLOC  ?X28[[PARM  ?X29[]]
                                                                         [RESTR  ?X30[[RELN  ?X3
1 既に-1]
                                                                                     [OBJE  ?X2
9]]]]]]]]]]]]]
               [CAT  ?X32 V]]]]
NIL

Command: (time (gen2 fs-2))
Evaluation of (GEN2 FS-2) took 566.458076 seconds of elapsed time
including 125.066 seconds waiting for the disk for 2,000 faults.
The garbage collector has flipped, so consing was not measured.
("登録用紙 を お持ち で しょ う か" "登録用紙 を お持ち で しょ う か")
```

付録4 日本語文生成結果

```
;;; -*- Syntax: Common-lisp; Default-character-style: (:FIX :ROMAN :NORMAL); Mode: LISP -*-

(setq fd-1 '((<0> == [[HEAD  [[FORM  FINITE]
                              [TENSE  PAST]
                              [mode declarative]]]
                      [TRANS  [[RELATION "sleep"]
                               [ARG1 [[CONDITION [[PARAMETER  ?X11]
                                                  [RESTRICTION [[ARG1  ?X11]
                                                                [RELATION  NAMED]
                                                                [ARG2  "uther"]]]]]
                                      [INDIVIDUAL  NIL]]]]]]])))


(setq fs-1 (car (make-prod-fss fd-1)))

(setq fd-2 '((<0> == [[HEAD [[FORM FINITE]
                             [TENSE PAST]
                             [mode declarative]]]
                     [TRANS [[relation perfective]
                             [arg1 [[RELATION "sleep"]
                                    [ARG1 [[CONDITION [[PARAMETER ?X11]
                                                       [RESTRICTION [[ARG1 ?X11]
                                                                     [RELATION NAMED]
                                                                     [ARG2 "uther"]]]]]
                                           [INDIVIDUAL  NIL]]]]]]]])))

(setq fs-2 (car (make-prod-fss fd-2)))

(setq fd-3
      '((<0> == [[HEAD [[FORM  NONFINITE]
                        [MODE  IMPERATIVE] ]]
                 [CAT  SENTENCE]
                 [TRANS [[ARG2 ?x09[[CONDITION [[RESTRICTION [[RELATION HEARER]
                                                              [ARG1  ?X13]]]
                                                [PARAMETER  ?X13]]]
                                    [INDIVIDUAL  NIL]]]
                         [ARG3 [[ARG2 [[condition [[RESTRICTION [[ARG1 [[CONDITION [[RESTRICTION
[[ARG1 ?X21]

 [RELATION "sword"]]]
                                                                                    [PARAMETER  ?
X21]]]
                                                                          [INDIVIDUAL NIL]]]
                                                                [RELATION "a"]
                                                                [ARG2  ?X25]]]
                                                  [PARAMETER  ?X25]]]]]
                                [RELATION  "use"]
                                [ARG1  ?X09]]]
                         [ARG1 [[CONDITION [[RESTRICTION [[RELATION SPEAKER]
                                                          [ARG1 ?X30]]]
                                            [PARAMETER ?X30]]]
                                [INDIVIDUAL NIL ]]]
                         [RELATION IMPERATIVE]]]] )))


(setq fs-3 (car (make-prod-fss fd-3)))

(setq fd-4
      '((<0> == [[HEAD [[FORM FINITE]
                        [tense present]
                        [MODE DECLARATIVE]]]
                 [CAT SENTENCE]
                 [TRANS [[ARG1 [[ARG1 [[CONDITION [[RESTRICTION [[RELATION  MALE]
                                                                 [ARG1 ?X15]]]
                                                   [PARAMETER  ?X15]]]
                                       [INDIVIDUAL NIL]]]
                                [RELATION "sleep"]]]
                         [RELATION "may"]]]])))

(setq fs-4 (car (make-prod-fss fd-4)))

(setq fd-5
      '((<0> == [[HEAD  [[FORM  FINITE]
                         [TENSE  PRESENT]
                         [MODE  DECLARATIVE]]]
                 [CAT  SENTENCE]
```

付録5 英文生成入力2 (1 of 2)

```
                [TRANS [[ARG2 [[CONDITION [[RESTRICTION [[RELATION MALE]
                                                        [ARG1  ?X15]]]
                                           [PARAMETER  ?X15]]]
                              [INDIVIDUAL NIL]]]
                       [RELATION  "buy"]
                       [ARG1 []]
                       [ARG3 [[CONDITION [[RESTRICTION [[ARG1 [[CONDITION  [[PARAMETER  ?X24]
                                                                            [RESTRICTION  [[ARG
1 ?X24]
                                                                                          [REL
ATION "sword"]]]]]]
                                                                 [INDIVIDUAL  NIL]]]
                                                       [RELATION  "a"]
                                                       [ARG2  ?X29]]]
                                                  [PARAMETER  ?X29]]]]]]]
      ]))))

(setq fs-5 (car (make-prod-fss fd-5)))

(setq fd-6
      '((<0> == [[HEAD  [[FORM  FINITE]
                         [TENSE  PRESENT]
                         [MODE  DECLARATIVE]]]
                 [CAT  SENTENCE]
                 [TRANS [[ARG2 [[CONDITION [[RESTRICTION [[RELATION MALE]
                                                         [ARG1  ?X15]]]
                                            [PARAMETER  ?X15]]]
                               [INDIVIDUAL NIL]]]
                        [RELATION  "buy"]
                        [ARG1 [[CONDITION [[PARAMETER ?X11]
                                           [RESTRICTION [[ARG1 ?X11]
                                                         [RELATION NAMED]
                                                         [ARG2 "uther"]]]]]
                              [INDIVIDUAL  NIL]]]
                        [ARG3 [[CONDITION [[RESTRICTION [[ARG1 [[CONDITION  [[PARAMETER  ?X24]
                                                                             [RESTRICTION  [[ARG
1 ?X24]
                                                                                           [REL
ATION "sword"]]]]]]
                                                                  [INDIVIDUAL  NIL]]]
                                                        [RELATION  "a"]
                                                        [ARG2  ?X29]]]
                                                   [PARAMETER  ?X29]]]]]]]
                 ])))

(setq fs-6 (car (make-prod-fss fd-6)))
```

付録5 英文生成入力2 (2 of 2)

```
Command: (time (gen2 fs-1))

Evaluation of (GEN2 FS-1) took 2395.717758 seconds of elapsed time
including 813.933 seconds waiting for the disk for 17,391 faults.
The garbage collector has flipped, so consing was not measured.
("uther slept")

Command: (time (gen2 fs-2))

Evaluation of (GEN2 FS-2) took 2697.983552 seconds of elapsed time
including 892.755 seconds waiting for the disk for 18,218 faults.
The garbage collector has flipped, so consing was not measured.
("uther had slept")

Command: (time (gen2 fs-3))

Evaluation of (GEN2 FS-3) took 1620.045961 seconds of elapsed time
including 534.540 seconds waiting for the disk for 10,001 faults.
The garbage collector has flipped, so consing was not measured.
("use a sword")

Command: (time (gen2 fs-4))

Evaluation of (GEN2 FS-4) took 6372.940443 seconds of elapsed time
including 2062.645 seconds waiting for the disk for 46,550 faults.
The garbage collector has flipped, so consing was not measured.
("he may sleep")

Command: (time (gen2 fs-5))

Evaluation of (GEN2 FS-5) took 5407.648061 seconds of elapsed time
including 1970.434 seconds waiting for the disk for 43,895 faults.
The garbage collector has flipped, so consing was not measured.
("he is bought a sword")

Command: (time (gen2 fs-6))

Evaluation of (GEN2 FS-6) took 5321.966213 seconds of elapsed time
including 1949.566 seconds waiting for the disk for 44,388 faults.
The garbage collector has flipped, so consing was not measured.
("uther buys him a sword")
```

付録6 英文生成結果2

```
Command: (time (gen2 fs-1))

Evaluation of (GEN2 FS-1) took 1233.775694 seconds of elapsed time
including 375.226 seconds waiting for the disk for 7,980 faults.
The garbage collector has flipped, so consing was not measured.
("uther slept")

Command: (time (gen2 fs-2))

Evaluation of (GEN2 FS-2) took 1246.934625 seconds of elapsed time
including 362.758 seconds waiting for the disk for 6,358 faults.
The garbage collector has flipped, so consing was not measured.
("uther had slept")

Command: (time (gen2 fs-3))

Evaluation of (GEN2 FS-3) took 691.407741 seconds of elapsed time
including 207.926 seconds waiting for the disk for 4,179 faults.
The garbage collector has flipped, so consing was not measured.
("use a sword")

Command: (time (gen2 fs-4))

Evaluation of (GEN2 FS-4) took 3171.964880 seconds of elapsed time
including 1029.792 seconds waiting for the disk for 20,981 faults.
The garbage collector has flipped, so consing was not measured.
("he may sleep")

Command: (time (gen2 fs-5))

Evaluation of (GEN2 FS-5) took 2556.921469 seconds of elapsed time
including 882.941 seconds waiting for the disk for 19,985 faults.
The garbage collector has flipped, so consing was not measured.
("he is bought a sword")

Command: (time (gen2 fs-6))

Evaluation of (GEN2 FS-6) took 2647.834568 seconds of elapsed time
including 920.911 seconds waiting for the disk for 22,007 faults.
The garbage collector has flipped, so consing was not measured.
("uther buys him a sword")
```

付録7 英文生成結果3

```
Command: (time (gen2 fs-1))

Evaluation of (GEN2 FS-1) took 67.193367 seconds of elapsed time including:
  2.213 seconds processing sequence breaks,
  11.783 seconds in the storage system (including 1.024 seconds waiting for pages):
    2.859 seconds processing 8422 page faults including 29 fetches,
    8.881 seconds in creating and destroying pages, and
    0.043 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("uther slept")

Command: (time (gen2 fs-2))

Evaluation of (GEN2 FS-2) took 67.951668 seconds of elapsed time including:
  2.452 seconds processing sequence breaks,
  11.030 seconds in the storage system (including 0.586 seconds waiting for pages):
    2.961 seconds processing 11403 page faults including 19 fetches,
    8.007 seconds in creating and destroying pages, and
    0.061 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("uther had slept")

Command: (time (gen2 fs-3))

Evaluation of (GEN2 FS-3) took 54.600697 seconds of elapsed time including:
  1.582 seconds processing sequence breaks,
  5.816 seconds in the storage system (including 0.179 seconds waiting for pages):
    2.233 seconds processing 9697 page faults including 7 fetches,
    3.556 seconds in creating and destroying pages, and
    0.027 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("use a sword")

Command: (time (gen2 fs-4))

Evaluation of (GEN2 FS-4) took 273.304470 seconds of elapsed time including:
  13.729 seconds processing sequence breaks,
  69.261 seconds in the storage system (including 29.622 seconds waiting for pages):
    39.323 seconds processing 33406 page faults including 908 fetches,
    29.775 seconds in creating and destroying pages, and
    0.164 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("he may sleep")

Command: (time (gen2 fs-5))

Evaluation of (GEN2 FS-5) took 1118.799100 seconds of elapsed time including:
  59.318 seconds processing sequence breaks,
  373.838 seconds in the storage system (including 254.208 seconds waiting for pages):
    294.827 seconds processing 119471 page faults including 7729 fetches,
    77.623 seconds in creating and destroying pages, and
    1.388 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("he is bought a sword")

Command: (time (gen2 fs-6))

Evaluation of (GEN2 FS-6) took 195.961380 seconds of elapsed time including:
  9.833 seconds processing sequence breaks,
  67.352 seconds in the storage system (including 43.868 seconds waiting for pages):
    50.360 seconds processing 19108 page faults including 1322 fetches,
    16.292 seconds in creating and destroying pages, and
    0.699 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("uther buys him a sword")

Command: (time (gen2 fs-5-1))

Evaluation of (GEN2 FS-5-1) took 648.510620 seconds of elapsed time including:
  34.154 seconds processing sequence breaks,
  220.452 seconds in the storage system (including 152.362 seconds waiting for pages):
    174.570 seconds processing 67908 page faults including 4398 fetches,
    44.844 seconds in creating and destroying pages, and
    1.037 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("he was bought a sword")
```

付録 8 英文生成結果 4 (1 of 2)

```
Command: (time (gen2 fs-6-1))

Evaluation of (GEN2 FS-6-1) took 143.266920 seconds of elapsed time including:
  5.540 seconds processing sequence breaks,
  38.747 seconds in the storage system (including 18.022 seconds waiting for pages):
    23.580 seconds processing 14832 page faults including 732 fetches,
    15.089 seconds in creating and destroying pages, and
    0.079 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("uther bought him a sword")
```

付録 8 英文生成結果 4 (2 of 2)

```
Command: (time (gen2 fs-1))

Evaluation of (GEN2 FS-1) took 1.916179 seconds of elapsed time including:
  0.055 seconds processing sequence breaks,
  0.496 seconds in the storage system (including 0.247 seconds waiting for pages):
    0.331 seconds processing 264 page faults including 18 fetches,
    0.165 seconds in creating and destroying pages, and
    0.000 seconds in miscellaneous storage system tasks.
54,619 list, 1,690 structure words consed in WORKING-STORAGE-AREA.
("uther slept")

Command: (time (gen2 fs-2))

Evaluation of (GEN2 FS-2) took 18.464146 seconds of elapsed time including:
  0.565 seconds processing sequence breaks,
  4.398 seconds in the storage system (including 0.117 seconds waiting for pages):
    0.594 seconds processing 2279 page faults including 3 fetches,
    3.792 seconds in creating and destroying pages, and
    0.012 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
NIL

Command: (time (gen2 fs-3))

Evaluation of (GEN2 FS-3) took 13.726389 seconds of elapsed time including:
  0.414 seconds processing sequence breaks,
  3.741 seconds in the storage system (including 0.127 seconds waiting for pages):
    0.512 seconds processing 1781 page faults including 5 fetches,
    3.228 seconds in creating and destroying pages, and
    0.001 seconds in miscellaneous storage system tasks.
195,961 list, 5,042 structure, 25,351 stack words consed in WORKING-STORAGE-AREA.
("use a sword")

Command: (time (gen2 fs-4))

Evaluation of (GEN2 FS-4) took 13.264420 seconds of elapsed time including:
  0.418 seconds processing sequence breaks,
  1.295 seconds in the storage system (including 0.015 seconds waiting for pages):
    0.289 seconds processing 1267 page faults including 1 fetches,
    1.007 seconds in creating and destroying pages, and
    0.000 seconds in miscellaneous storage system tasks.
168,108 list, 5,369 structure, 59,230 stack words consed in WORKING-STORAGE-AREA.
("he may sleep")


Command: (time (gen2 fs-5-1))

Evaluation of (GEN2 FS-5-1) took 120.649239 seconds of elapsed time including:
  4.944 seconds processing sequence breaks,
  29.304 seconds in the storage system (including 13.732 seconds waiting for pages):
    17.766 seconds processing 14852 page faults including 478 fetches,
    11.489 seconds in creating and destroying pages, and
    0.048 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("he was bought a sword")

Command: (time (gen2 fs-6-1))

Evaluation of (GEN2 FS-6-1) took 20.860437 seconds of elapsed time including:
  0.638 seconds processing sequence breaks,
  2.580 seconds in the storage system (including 0.908 seconds waiting for pages):
    1.498 seconds processing 2361 page faults including 61 fetches,
    1.081 seconds in creating and destroying pages, and
    0.001 seconds in miscellaneous storage system tasks.
The garbage collector has flipped; so no consing was measured.
("uther bought him a sword")
```

付録9 英文生成結果5