英文添削のための対話型システム ASPEC-II†

河合敦夫†† 杉原厚吉†† 杉江昇††

本論文では，人間と計算機システムが対話しながら英文の文法的誤りをみつけてゆくシステム ASPEC-II について述べる．英文の文法的な誤りをみつける手法の一つとして，補強文脈自由文法を用いることができる．しかし，すべての英文を正しく処理できる補強文脈自由文法を用意することはむずかしい．また，種々の文法的誤りを，すべて補強文脈自由文法で検出するのもむずかしい．そこで，ASPEC-II では，システムのもつ文法規則で構文解析できないときは，システムとユーザが対話を始める．対話のために，ユーザがシステムに入力文の骨組を教えるコマンドや，システムが認識している部分解析木をユーザに表示するコマンドなどが用意されている．ユーザは，対話によって得た情報をもとに，構文解析できない原因を判断する．英文に誤りがあると判断すれば，コマンドを使って修正する．システムの文法が不足していると判断すれば，これを追加する．この手法により，著者らが以前に開発したシステム ASPEC-I より広範囲の文法的誤りが検出でき，使いやすさも向上した．

1. まえがき

我々が書く文章にはつねに正しいとは限らない．そのなかには，綴りの誤りや文法的な誤り，さらには意味上の誤りなど，いろいろな誤りがある．こうした誤りを，計算機がみつけてくれれば便利である．

英文の誤りを検出する計算機システムについて考えてみる．まず，単語の綴りの誤りを検出するシステムは，すでに実用化されている[1]．また，Unix のシステム[2),3)]では，綴りの誤りのほかに，文体上の特徴も指摘する．計算機による自然言語の処理技術からみて，その次に位置する段階として，文法的な誤りの検出がある．これについては，最近，試作システム[4),5)]が作られている．

これらのシステムは，誤り検出の手法として，補強文脈自由文法を用いている（この手法については，2.1節で簡単に述べる）．しかし，2.2節で述べるように，この手法だけで多くの種類の文法的な誤りを検出することはむずかしい．

そこで，我々は，人と計算機システムが対話しながら誤りをみつけてゆくシステム，ASPEC-II (an Advisory System for Polished English Composition II) を作成した．その概要を2.3節に，具体的な処理方法を3章に，コマンドの使用例を4章で述べる．また，ASPEC-II を用いて行った実験の結果を5章で述べる．

† ASPEC-II: An Interactive Error Detection System for English Composition by ATSUO KAWAI, KŌKICHI SUGIHARA and NOBORU SUGIE (Department of Information Science, Faculty of Engineering, Nagoya University).

†† 名古屋大学工学部情報工学科

2. 補強文脈自由文法による誤り検出とその問題点

2.1 補強文脈自由文法による誤り検出

補強文脈自由文法[6)]は，一つの記号が他のどんな記号に置きかえられるかという規則（生成規則）と，その生成規則をどんなときに適用するかを決める部分（生成規則の制御部分）から成りたっている．この補強文脈自由文法を使って英文を解析すると，たとえば**図1**に示す構文解析木ができる．

図1の英文には，冠詞の誤り［“a” を “an” に修正すると正しい］がある．この誤りは，生成規則［NSTG →LN＋＃NSTG］に，制御部分［名詞句の一部である＃NSTG の先頭の単語が母音で始まるとき，冠詞を表すLN を調べよ．“an” であれば，生成規則を適用する．“a” であれば，エラーメッセージの登録をしてから，生成規則を適用せよ］を加えると，誤り検出ができるようになる．

2.2 補強文脈自由文法による誤り検出の問題点

2.1節で述べた方法を使うと，数の不一致の誤り（I bought this two fish.: these が正しい）や，形容詞の用法の誤り（It is an alive snake.: a live などが正しい．叙述的に用いる形容詞を限定的に用いた誤り）などをはじめとして，いろいろな文法的誤りが検出できる[4),5)]．しかし，この方法では，適切な構文解析木を作ることができて初めて誤りの検出が可能となる．そのため，この方法だけですべての誤りを検出しようとすると，次の(1)～(3)で述べるシステム側の文法での問題点と，(4)で述べる入力文における問題点がでてくる．

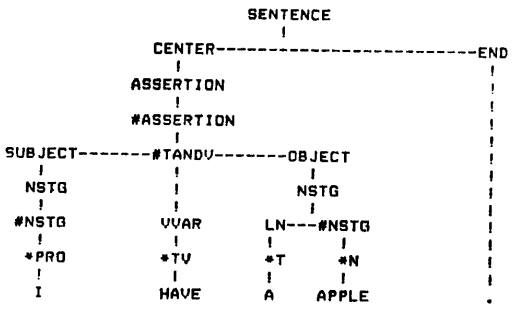

図 1 構文解析木の例
Fig. 1 An example parse tree.

(1) 英文が文法的に正しいと仮定しても，入力されるどんな英文にたいしても適切な構文解析木を得ることができる文法のセット（生成規則とその制御部分の集まり）を作成するのは，なかなかむずかしい．著者ら[4]が行った実験では，2進木（1個の記号が，たかだか2個の記号に分かれる様式）の生成規則を 402 個用意した場合，Scientific American の英文にたいして，65% の割合で解析が可能であった．また，より大規模な文法[7),8)]が作られてはいるものの，それらの文法でも，すべての例外的な英文を解析できるわけではない．

(2) 生成規則を増加させるのは，容易ではない．

これは，次の理由による．正しい文を解析できるようにするために，生成規則を追加したとする．しかし，この追加した生成規則のために，文法的な誤りが原因で解析木が作れない文において，(書き手の意図しない) 適切でない解析木ができる危険性がある．これでは，誤りのある文も正しいと判定されてしまう．これは，追加した生成規則の制御部分の記述が不十分であることが原因である．もちろん，制御部分の記述を，あらゆる英文を想定して記述すればよいのだが，これは非現実的である．

(3) また，生成規則を増加させてゆくと，計算機による処理時間や記憶容量が実用の域を越えてしまう．

(4) 入力文は，いろいろなタイプの文法的誤り (たとえば，動詞の書き忘れ) を含んでいる．こうした誤りのために，構文解析木ができないことがある．このときは，構文解析不可能というメッセージしかシステムは出さないから，英文の誤りがどこにあるかに関する情報は，まったく得られない．すなわち，2.1 節の手法では，解析木を作ることができて，初めて誤りの検出ができる．これでは，システムとして不完全である．

もちろん，いろいろなタイプの誤り英文の解析のために文法を作成すれば，理論的には解析木を得ることができる．しかし，(1)～(3)で述べたように，正しい英文でさえ完全な文法を用意するのはむずかしい．したがって，誤りを含む英文ではさらにむずかしい．

(1), (2)の難点を克服するために，文献 9) では，"fitting procedure" と呼ばれる方法を提案している．この方法は，システム側の補強文脈自由文法だけで解析できないとき，ヒューリスティクな手法をシステムが自動的に用いて，近似的な解析木を得る方法である．しかし，こうした，処理を自動的に行う方法では，書き手の意図を伝えられないし，解析できなかった原因も直接にはわからない．

そこで，この問題を克服するために，人とシステムが対話する機能をもつシステム，ASPEC-II を作成した．

なお，対話機能を組み込んだシステムとしては，文献 10) がある．このシステムは，文脈自由文法だけでなく，手続的な解析規則を組み合わせたシステムであるが，英文の誤りの検出を目的としたものではない．

2.3 ASPEC-II の概要

ASPEC-II は，補強文脈自由文法による誤り検出システム ASPEC-I に，対話機能を追加したシステムである．

ASPEC-II の英文処理の流れを図 2 に示す．まず

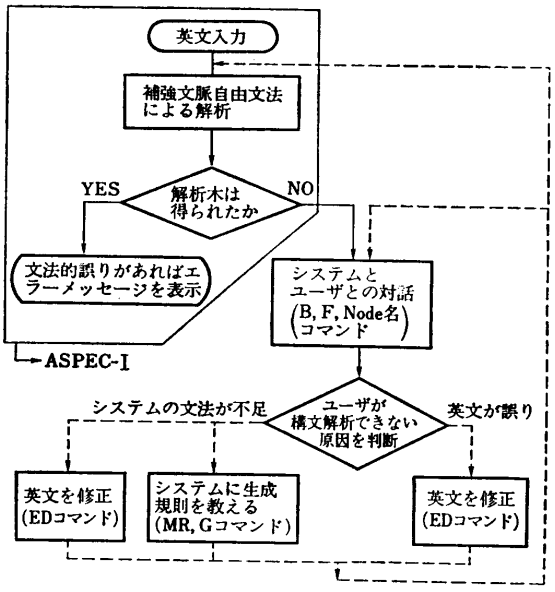

図 2 ASPEC-II の概要
Fig. 2 Flow chart in ASPEC-II.
⟶ システムによる処理，⟶ ユーザのコマンドによる処理

入力文の構文解析を試み，解析木が得られた場合は，その解析木をもとにして誤りの有無を判定する．この部分が，我々が以前に試作したシステム ASPEC-I である．一方，構文解析ができなかった場合には，システムとユーザとの対話をコマンドを使って行う．対話により，英文が誤っているとユーザが判断すれば，コマンドを使って英文を修正する．システムの文法が不足していると判断すれば，コマンドにより文法を補足するか，英文に語句を補ったり削除したりして，解析できるようにする．

3. 対話コマンドとその処理機能

本章では，ASPEC-II に用意されているコマンドとその機能について述べる．

3.1 Node 名コマンド

英文の書き手がどんな文型で英文を書こうとしたかを計算機システムに教えたいとする．その一つの方法として，たとえば「この部分を前置詞句のつもりで書いた.」といった教え方をすることができる．このためのコマンドが，Node 名コマンドである．

Node 名コマンドは，(Node 名 No. 1 No. 2) の形式でシステムに入力する．これにより，先頭より No. 1 番目の単語から，No. 2 番目までの単語列が，Node 名からなるまとまりであることを教える．たとえば，(Subject 1 5) と入力すると，先頭の単語から5番目までの単語列が主語であると教えたことになる．ここで，Node 名として使うことのできる記号として，次の2種類がある．(1)システム側の生成規則を作るときに使われている記号（例：NSTG（中心となる語が名詞である名詞句））(2)一般的に使われる記号であるが，(1)に含まれない記号 12 個（例：名詞句を表す NP（この記号を入力すると，システムが自動的に，VINGO（動名詞），TOVO（不定詞），NSTG のいずれかであるとみなす））

さて，Node 名コマンド (Node 名 No. 1 No. 2) を入力すると，システム側は次の(1)，(2)の条件を満たす部分解析木（文の一部の単語列から構成した解析木）を作ろうとする．(1) No. 1 番目から No. 2 番目までの単語列から構成されている．(2)解析木の一番上の根にあたる部分の記号（今後，解析木の初期記号と呼ぶ）が，Node 名と一致する．

もし，部分解析木を作ることができれば，ユーザが教えたことと，システム側の認識が一致したことになる．このときは，システムが部分解析木を記憶して文全体の解析のときに利用する．部分解析木を作ることができないときは，ユーザとシステムの認識が異なっていることを，ユーザに表示する．この表示により，システムが文全体の構文解析に失敗した原因は，現在注目している部分単語列のなかにある一ことをユーザが知ることができる．

しかし，一般には，書き手の意図する文型をシステム側に教えるためには，何度も Node 名コマンドを使わなければならない．これでは面倒なので，次の方法を考える．

3.2 B (bottom up) コマンド

Bコマンドは，システムの文法を使って作ることのできる部分解析木をユーザに表示するためのものである．

このコマンドを入力すると，システムは，ユーザの指示した単語列から得られるすべての部分解析木を，ボトムアップに求める．求まった解析木を，ユーザに見やすい形で，単語数の多い順に表示する．この表示を見れば，システムが入力文をどのように認識しているかがわかる．たとえば，ユーザが名詞句のつもりで書いた単語列が，システムにより名語句として表示されていなかったとする．そうすると，文全体の構文解析ができなかった原因の一つが，その単語列であることがわかる．また，システムが表示するいくつかの部分解析木のなかから，ユーザが適切と判断する解析木を選ぶこともできる．このときは，選んだ解析木をシステムが記憶しておき，文全体の解析のときに利用する．

なお，ボトムアップだけの解析は，トップダウンの解析を併用したときに比べて，解析時間が一般に長く

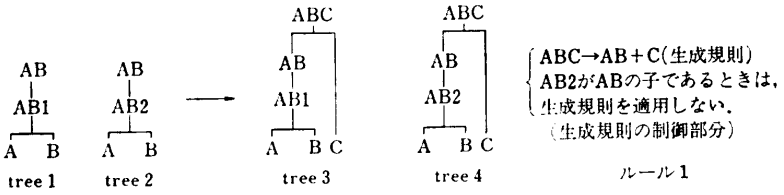

図 3 解析木が一つも求まらない例
Fig. 3 An example in which no parse tree is obtained.

かかる．そこで，計算時間や記憶容量を減らすために，単語数の多い英文では，次の方法を採用する．

すなわち，解析の途中で，同一の単語列にたいして複数の解析木が生じたとする．このとき，解析木の初期記号が同じであれば，複数の解析木のうちから一つを選択する方法である．

しかし，この方法には欠点がある．これは，この方法を使わないときには求まっていた解析木が，使うことにより求まらない危険性があるからである．これは**図3**に示す仮想的な例で説明できる．解析木を一つに絞らない方法では，解析の最終段階で tree 3 と tree 4 が残る．そして，図3のルール1により，tree 3 が残る．ところが，解析木を一つに絞る方法では，tree 1 と tree 2 が並列的に作られた時点で，どちらかの木が選択される（二つの解析木の初期記号が同じであるから）．ここで，tree 2 が選択されたとする．すると，最終段階で tree 4 を作ろうとするが，ルール1により禁止され，どんな木も残らない．

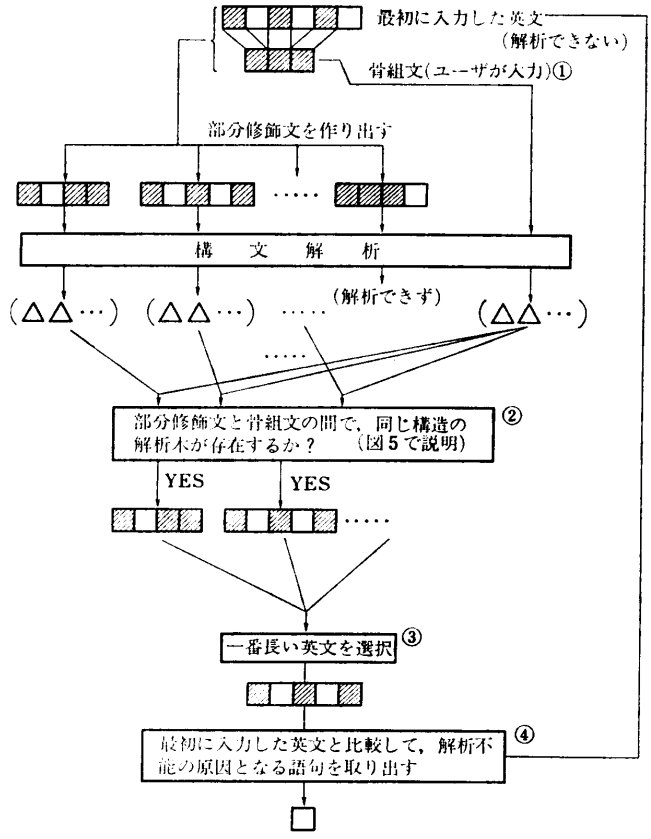

図4 Fコマンドによる英文処理
Fig. 4 Sentence processing by F command.
▭, ▨: 英文の語句, ▭: 修飾語句，挿入語句, △: 構文解析木

したがって，Bコマンドで，ユーザが意図する解析木がもしみつからないときは，Node 名コマンドで再確認する必要がある．

この方法を採用することにより，Bコマンドによる処理時間を大幅に減らすことができた．

3.3 F (framework) コマンド

3.1 節，3.2 節で述べたコマンドは，単語のまとまりとしての句や節などの名前（たとえば名詞句）を使って，システムとの対話を行っている．しかし，句や節などの名前は，文法の作り方によりいろいろな呼び方がある．したがって，同じ句にたいする呼び方が，ユーザとシステム側で異なることもある．このときは，書き手の意図をシステムに教えにくい．

そこで，英文の骨組を入力することによって，書き手の意図する文型をシステムに教えるためのものがFコマンドである．

Fコマンドの英文処理の流れを**図4**に示す．まず，解析できなかった英文から修飾語句や挿入語句を取り除いた文（以後，骨組文と呼ぶ）を，ユーザが入力する（図4の①）．システムは，骨組文に修飾語句や挿入語句の一部を肉付けした文（以後，部分修飾文と呼ぶ）を作り出す．そして，それぞれの部分修飾文が，ユーザの入力した骨組文と同じ構造をしているかを判定する（図4の②）．同じ構造をしている部分修飾文のなかから，一番長い英文を選択する（図4の③）．それを，解析できなかった英文と比較することにより，最初に入力した英文が解析できない原因となる修飾語句や挿入句がみつかる．

また，このようにしてみつけた修飾語句や挿入句の長さが長いときには，さらに，その骨組となる語句を入力して，同様の処理を行うこともできる．

ところで，図4における，二つの解析木が同じかどうかの判定は，つぎの判定手順により決める．

〔判定手順〕

図5は，この判定に使う，解析木の構造の分類である（2進木の場合）．[type 4] および [type 5] のなかで，非終端記号 C を根とする部分解析木が，挿入語句や修飾句を表す．

［1］ type 1 であれば，同じ構造であると判定する．

［2］ type 2 であれば，tree O と tree M

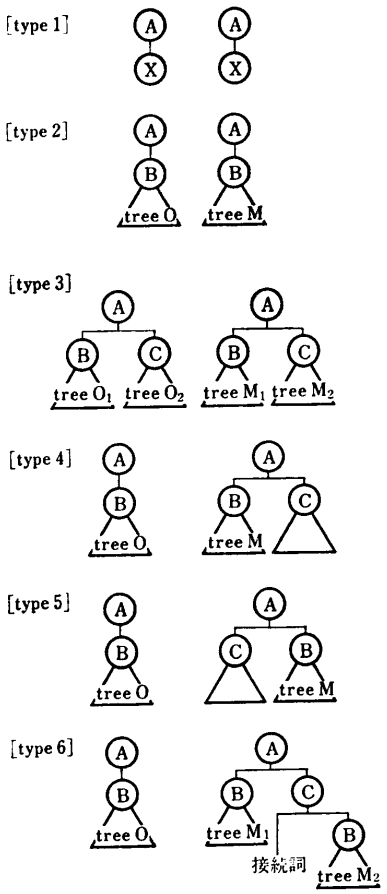

図 5 判定のための解析木の分類
Fig. 5 Types of parse trees.
左側が骨組文，右側が部分修飾文，X：終端記号，
A, B, C：非終端記号

が同じ構造であるかを判定する．

［3］ type 3 であれば，tree O_1 と tree M_1，tree O_2 と tree M_2 がそれぞれ同じ構造であるかを判定する．

［4］ type 4 であれば，tree O と tree M が同じ構造であるかを判定する．

［5］ type 5 であれば，tree O と tree M が同じ構造であるかを判定する．

［6］ type 6 であれば，tree O と tree M_1 または，tree O と tree M_2 が同じ構造であるかどうかを判定する．

［7］［1］～［6］の判定を再帰的に適用し，どこかで異なると判定されれば，全体の構造が異なると判定する．

［8］［1］～［7］以外であれば，同じであると判定する．

この判定で十分であるかどうかは，生成規則の作り方による．ASPEC-II では，この判定法で十分であるように，生成規則を作成している．

3.4 MR (make a rule) コマンド

システムの文法が不足しているために英文が解析できないと，ユーザが判定したときに，生成規則をシステムに追加するコマンドである．

生成規則（A→B+C）と，非終端記号 B，C が，それぞれどの単語列に相当するかを教える．教えた生成規則はシステムに仮登録され，現在解析中の英文を処理するときにだけ使われる．

ここで登録した生成規則を，一般的なルールとしてシステムの中に取り入れ，別の英文の解析に使うのは危険である．なぜなら，生成規則の適用を制御する部分の記述を十分吟味しないまま，他の英文の解析に使うと，書き手の意図しない形の解析木ができる危険性があるからである．

3.5 G (grammar) コマンド

MR コマンドでは，ユーザが生成規則を教える必要がある．これでは，ユーザが適切な生成規則を思いつかないときには困る．

Gコマンドでは，ユーザが，単語列と一つの非終端記号を入力する．システムは，この単語列を使って，非終端記号を根（初期記号）とする部分解析木を作る．

3.6 ED (edit) コマンド

ED コマンドは，単語の挿入，削除，置換の操作によって英文の修正をするときに使う．これは，英文の誤りを訂正するときや，文法が不足して英文が解析できないときに語句を補ったり削除して解析できるようにするときに使う．修正の終了した英文が解析できるかどうかを，再びシステムを使って調べる．

4. コマンドの使用例

4.1 B, ED コマンドの使用例

誤りのある英文を，ASPEC-II によってみつけ，修正する例を**図 6** に示す．

① の入力文は構文解析できなかったので，B コマンドを使って原因を調べる（②）．B コマンドの出力結果が ③ である．簡単のため，得られた部分解析木の初期記号 (assertion, center, …) および単語列のみを表示している．③ から，入力文は，③-1 (The image…) と，③-4 (The television camera…) の二つの平叙文 (assertion) が混ざっていることがわかる．

```
① (THE TELEVISION CAMERA DIVIDES THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN LINES . )
   1   2          3      4       5   6     7          8  9 10     11 12     13 14   15          16   17    18
② ENTER-COMMAND=(B 1 17)
  (THE TELEVISION CAMERA DIVIDES THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN LINES )
③  1(THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN LINES )
     (ASSERTION CENTER THATS-THAT SN)
   2(IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN LINES )
     (ASSERTION CENTER THATS-THAT SN)
   3(THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN )
     (ASSERTION CENTER THATS-THAT SN)
   4(THE TELEVISION CAMERA DIVIDES THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES )
     (ASSERTION CENTER THATS-THAT SN)
  _GOOD-NO=NIL
④ ENTER-COMMAND=(ED)
   (THE TELEVISION CAMERA DIVIDES THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES IS MANY HORIZONTAL SCAN LINES . )
    1   2          3      4       5   6     7          8  9 10     11 12     13 14   15          16   17    18
⑤ _SUB-COMMAND=(R 13 INTO)
⑥  (THE TELEVISION CAMERA DIVIDES THE IMAGE CONSTITUTED BY A SYSTEM OF LENSES INTO MANY HORIZONTAL SCAN LINES . )
    1   2          3      4       5   6     7          8  9 10     11 12     13   14   15          16   17    18
  _SUB-COMMAND=NIL
  ENTER-COMMAND=NIL
  +ALL-PARSE#1: NIL
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  "THE FOLLOWING ERRORS ARE FOUND."
⑦ NIL
```

図 6 ASPEC-II の使用例 (B, ED コマンド)
Fig. 6 The use of ASPEC-II (B, ED commands).

```
① (THE NUMBER , BESIDES BEING A RATHER UNINFORMATIVE STRING OF DIGITS , IS THE SAME AS THE NUMBER OBTAINED FROM
   THE NUMERICAL EVALUATION OF 1973 . )
    1   2      3 4       5     6 7      8              9      10 11     1213 14  15   16 17  18     19       20
   21  22       23        24 25   26
② ENTER-COMMAND=(F 1 25)
  (THE NUMBER , BESIDES BEING A RATHER UNINFORMATIVE STRING OF DIGITS , IS THE SAME AS THE NUMBER OBTAINED FROM
  THE NUMERICAL EVALUATION OF 1973 )
③ ENTER-ESSENTIAL-SENTENCE=(THE NUMBER IS THE SAME /.)
④ ESSEN-SENTENCE=(THE NUMBER IS THE SAME AS THE NUMBER OBTAINED FROM THE NUMERICAL EVALUATION OF 1973 . )
⑤   1(. BESIDES BEING A RATHER UNINFORMATIVE STRING OF DIGITS , )
  SEARCH-NO=NIL
⑥ ENTER-COMMAND=(B 4 11)
  (BESIDES BEING A RATHER UNINFORMATIVE STRING OF DIGITS )
⑦  1(BEING A RATHER UNINFORMATIVE STRING OF DIGITS )
     (VINGO SUBJECT SA)
   2(A RATHER UNINFORMATIVE STRING OF DIGITS )
     (NPN NSTG SUBJECT PASSOBJ)
  _GOOD-NO=NIL
⑧ ENTER-COMMAND=(MR 4 11)
  (BESIDES BEING A RATHER UNINFORMATIVE STRING OF DIGITS )
⑨ _ENTER-LEFTWORDS=(4 4)
⑩   0  LP ->DSTG
    1  SA ->DSTG
    2  DSTG ->*D
    3  *P ->BESIDES
⑪ _GOOD-NO=3
⑫ _ENTER-RIGHTWORDS=(5 11)
⑬   0  SA ->VINGO
    1  SUBJECT ->VINGO
    2  VINGO ->#VINGO
⑭ _GOOD-NO=2
⑮ _ENTER-NODE-NAME=PN
⑯ _PRODUCTION-RULE=  PN ->(*P . VINGO)
  ENTER-COMMAND=NIL
  +ALL-PARSE#1: NIL
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  "THE FOLLOWING ERRORS ARE FOUND."
⑰ NIL
```

図 7 ASPEC-II の使用例 (F, B, MR コマンド)
Fig. 7 The use of ASPEC-II (F, B, MR commands).

ユーザが，エディタ (④) の置換 (=Replace, ⑤) コマンドを使って英文を修正する．修正後の英文(⑥)を，再び構文解析すると，誤りのないことがわかる(⑦)．

4.2 F, B, MR コマンドの使用例

図 7 は，システムに文法を追加することにより，解析ができるようになる例である．

入力文①が解析できなかったので，Fコマンド(②)を使って原因を調べる．まず，ユーザが骨組文③を入力する．すると，システムは，部分修飾文④は認識できるが挿入句⑤が原因で全文①の解析ができないこと，を表示する．そこで，B コマンド (⑥) を使い，挿入句⑤をシステムがどのように認識しているかを調べる．すると，システムは認識している部分解析木を簡略化して表示する (⑦)．⑦ から，動名詞 (VINGO) は認識しているが，"besides being～digits" を前置詞句 (PN) として認識していないことがわかる．

そこで，生成規則を追加する MR コマンド (⑧) を入力する．システムは，"besides" を前置詞句を修飾 (LP) する副詞句 (DSTG) : (例) *just* before noon, 文全体を修飾 (SA) する副詞句 : (例) He left *soon*,

```
①(THIS MEDICINE CAN CURES YOUR COUGH . )
  1    2        3   4    5    6     7
②ENTER-COMMAND=(SUBJECT 1 2)
 (THIS MEDICINE )
   0  SUBJECT ->NSTG
 _GOOD-NO=0
 ENTER-COMMAND=NIL
 +ALL-PARSE#1: NIL
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
③"THE FOLLOWING ERRORS ARE FOUND."
 (ERROR1)
  (CHANGE (CURES) TO ROOT-FORM)
```

図 8 ASPEC-II の使用例 (Node 名コマンド)
Fig. 8 The use of ASPEC-II (Node command).

副詞句 (DSTG), 前置詞 (*P) などと認識している (⑩). ユーザは, 前置詞のつもりで書いたので, No. 3 を選択する (⑪). また, "being～digits" は, 動名詞 (VINGO) のつもりで書いたので, No. 2 を選択する. 新しい生成規則 ⑯ を使うことにより, 全文の解析が可能となる (⑰).

4.3 Node 名コマンドの使用例

英文の書き手の意図をシステムに伝えることにより, 誤りがみつかる例を図 8 に示す.

図 8 の英文 ① では, 少なくとも二つの解析木を作ることができる. (1) (This medicine can) (cures your cough). この薬びんは, …(2) (This medicine) (can cures your cough). この薬は, …. もし, 書き手が(2)の意図で書いたとすれば, 文法的に誤りがあることになる. ところが, このことをシステムに伝えないと, 文法的には誤りのない(1)の解析が可能なので, 誤りは検出できない. そこで, 書き手の意図を Node 名コマンドを使ってシステムに教える (②) と, 誤りの検出ができるようになる (③).

5. 実験結果

5.1 誤りのある文の場合

ASPEC-II の対話機能を使って, 誤りをどれくらいみつけることができるかについて述べる. 大学院生等 8 人が書いた合計 71 個の英文のうち, 7 文に文法的な誤りのあることが, ネイティブスピーカにより発見された. このうち, 5 文については, 構文解析木を作ることができた. この 5 文中, 4 文については, システムが自動的にエラーメッセージを表示した (文献 4) 参照). 5 文中, 残りの 1 文は, 「現在」の時制で書くべき文を, 「過去」の時制で書いた誤りである. この誤り検出には, 前後の文の情報 (文脈情報) が必要なため, 誤りは検出できなかった. 7 文中, 残りの 2 文については, 文法的な誤りのために構文解析木を作ることができなかった. この場合, ASPEC-II の対話コマンドを使うことにより, 解析できない原因となる語句 (この場合は, 文法的に誤りのある語句) をみつけることができた. その一つは, すでに図 6 に示した. どのように修正すると良いかまでは, システムは表示することができない.

表 1 使用したコマンドの統計 (単位は, コマンド/文)
Table 1 Summarized table of employed commands.

コマンド	ユーザ 最初の18文	ユーザ 最後の18文	システムの作成者
Node 名			
center	2.5	2.2	0.6
PN	0.2	0.6	0.2
NSTG	0.3	0.2	0.4
その他	0.2	0.5	0.5
B (bottom up)	1.2	0.1	0.7
F (framework)	0.0	0.0	0.1
ED (edit)	0.9	1.3	0.3
G (grammar)	0.4	0.0	0.8
MR (make a rule)	0.0	0.0	0.5
合 計	5.7	4.9	4.1

5.2 コマンドの使いやすさ

ASPEC-II のもっている文法や, 解析手法について知識のないユーザが, どれくらいコマンドを使うことができるかについて, 実験を行った. Scientific American および情報処理学会誌の英文アブストラクトから, ランダムに選んだ合計 200 文の英文のうち, ASPEC-II のもつ文法で自動的に解析できなかった 46 文を対象とした. この 46 文について, ASPEC-II の対話コマンドを使うことにより, 解析できない原因を判断し, 英文を添削したり, 文法を追加したりして, なるべく解析ができるようにすることを課題とした. この課題のために, ユーザとシステムの作成者が使ったコマンドの統計を表 1 に示す.

解析できない原因となる語句をみつけるために, B, Node 名, F コマンドを使うことができる. このうち, ユーザが最も多く使ったコマンドは, Node 名コマンドである. Node 名コマンドは, 3.1 節で述べたように, いろいろな使い方ができる. たとえば, 先頭から No. 1 番目の単語から No. 2 番目の単語までが, 一つの文であるかどうかを調べるには, (Center No. 1 No. 2), 前置語句かどうかは (PN No. 1 No. 2) と入力する. (center No. 1 No. 2) と入力することにより, たとえば重文や複文において, どの文が原因で解析できないかを, 大局的に調べてゆくことができる. Bコマンドは, 最初は多く使われた. しかし,

3.2 節で述べたように，すべての解析木が表示されない，などの理由で，後半では使われなくなった．

英文がシステムで解析できるようにするためには，ED，G，MR コマンドを使うことができる．ED コマンドが多く使われている．ED コマンドを使うときには，何らかの語句を補ったり，取り除いたりして，解析できるようにする．また，Gコマンドは，システムが処理できなかった慣用句等を，強制的にシステムに登録（たとえば名詞句として）するために使われている．生成規則を教える MR コマンドは，ユーザによって使われていない．システムの文法をまったく知らないユーザが，生成規則をシステムに教えるのは，むずかしいためである．

全体として，1文あたりのコマンド使用数は，慣れてくると減ってくることがわかる．

ユーザが解析できない原因となる語句をみつけることは，多くの場合（76%）に可能であった．原因をみつけることのできなかった 11 文（24%）のうち，Scientific American の4文については，システムの文法が十分でなかったためである．また情報処理学会誌の英文アブストラクトでは，システムの文法が不十分であるもの1文，動詞がホーンビーの動詞パターン[11]にない使われ方をしているもの3文，著者の英語能力からではシステムの文法が不十分なのか，英文が一般的でないのか，判明しなかったもの3文であった．

図6～図8に示す程度の英文においてコマンドを使うことができるのであれば，もともと正しい英文が書けて，添削システムは不要であるとも，考えられる．しかし，ユーザは，自分の書いた英文を対象としてシステムを使うから，自分の書いた英文は理解しているはずであり，それぞれの人の英語力に応じてシステムを使うことができる．また，英語の能力がある人でも，ケアレスミスのような誤りをすることがある．こうした誤りについても，本システムは有効である．

5.3 システムの解析速度

ASPEC-I での英文1文（単語数は平均 20.4 語／文）あたりの処理速度は，平均して約3秒[4]である．ASPEC-II の解析手法は，ASPEC-I と同じなので，解析時間はインターラクティブな場合も含めて，同程度である．ただし，3.2 節において述べたように，Bコマンドでは，ボトムアップの手法による解析のため，処理時間が長くかかる．これについては，今後の改良がさらに必要である．

5.4 まとめ

実験の結果をまとめると，まず，解析できない原因となる語句をみつけるためには，ASPEC-II のコマンドは有効であった．また，システムの知識のないユーザが，英文を解析できるようにするために使うコマンドのなかには，使い勝手が不十分なものもある．

6. むすび

計算機システムと人間との対話により，文法的な誤りをみつけてゆくシステム，ASPEC-II について述べた．このシステムは，補強文脈自由文法を使った誤り検出の方法では検出しにくい種類の誤りも検出できる点，システム側の文法が不十分であっても解析が行える点に特徴がある．

また，ASPEC-II は，システム自身の文法を改良してゆく上においても役に立つ．著者自身も，ASPEC-II により，以前に比べて簡単に，システムの文法を改良できた．

今後は，ユーザとシステムとの対話を，より使いやすい形にしてゆくことが必要である．

本研究の一部は，文部省科研費（課題番号 57580022）により行われ，プログラムの実行には，名古屋大学大型計算機センターを利用した．

参考文献

1) 伊藤哲郎：英文つづり誤りの訂正法，情報処理，Vol. 25, No. 5, pp. 471-479 (1984).
2) Macdonald, N. H.: The UNIX™ Writer's Workbench Software: Rationale and Design, *Bell Syst. Tech. J.*, Vol. 62, No. 6, pp. 1891-1908 (1983).
3) Gringrich, P. S.: The UNIX™ Writer's Workbench Software: Results of a Field Study, *Bell Syst. Tech. J.*, Vol. 62, No. 6, pp. 1909-1921 (1983).
4) 河合，杉原，杉江：英文の誤りを検出するシステム ASPEC-I，情報処理学会論文誌，Vol. 25, No. 6, pp. 1072-1079 (1984).
5) Heidorn, G. E., Jensen, K., Miller, L. A., Byrd, R. J. and Chodrow, M. S.: The EPISTLE text-critiquing system, *IBM Syst. J.*, Vol. 21, No. 3, pp. 305-327 (1982).
6) 伊理正夫：数と式と文の処理，p. 230，岩波書店，東京 (1981).
7) Sager, N: *Natural Language Information Processing*, Addison-Wesley, Reading (1981).
8) Robinson, J. J.: DIAGRAM: A GRAMMAR FOR DIALOGUES, SRI Technical Note 205 (1980).

9) Jensen, K., Heidorn, G. E., Miller, L. A. and Ravin, Y.: Parse Fitting and Prose Fixing: Getting a Hold on Ill-formedness, *Am. J. Comput. Linguist.*, Vol. 9, Nos. 3-4, pp. 147-160 (1983).
10) 西田，堂下：自然言語解析のためのプログラミングシステム COMPLAN について，情報処理学会論文誌，Vol. 23, No. 4, pp. 396-405 (1982).
11) ホーンビー，A. S.（伊藤健三訳）：英語の型と語法，オックスフォード大学出版局，東京(1977).

(昭和59年9月 4 日受付)
(昭和60年2月21日採録)