3M−8　上昇型チャート法に基づく適切な構文誤り解析手法

加藤裕史　武田正之　井上謙蔵
(東京理科大学)

1．はじめに

従来の構文誤り解析処理（以下，単に誤り処理と呼ぶ）では，もっとも解析が進んだ箇所を誤り箇所とし，その箇所から予想される事柄によって誤り原因を指摘するのが一般的であった．

しかし，この方法が本当に適切な誤り処理といえるのであろうか？

例えば，次のようなPascalの文を考えてみよう．

… ; x := 1 to 5 do …

これは，明らかに，";"の右隣に予約語"for"が欠けているために発生する構文誤りである．しかし，従来の誤り処理手法では"x := 1"を代入文の1部として解釈してしまい，「予約語"to"が予想されていない位置に存在する」といった診断を下してしまう．このことは，ある程度の知識を持つ人間なら容易に指摘できるような単純な誤りさえ，従来の手法では適切に指摘できないことを示している．

では，どうしたらもっと適切な誤り処理を行なうことができるのであろうか？

今，人間がプログラムをデバッグする場合について考えてみよう．このとき，プログラムがおかしいと感じた場合，彼はそのおかしいと思われる部分だけから誤りを診断するだろうか？おかしいと思われる部分の周辺を，必要な範囲において，もう1度眺めてから，初めて誤りを診断するのが普通だろう．その方が真の誤り原因を発見しやすくなるからである．これは別に何もプログラム言語に限ったことではない．例えば，人間が自然言語からなる文を推敲する場合においても同様である．

このような手法を計算機上で実現できないだろうかと考えついたのが，今回発表する新しい誤り処理手法である．我々は，まずこの手法を部分統制チャート法（以下，単にチャート法と略す）[Kay 80]に基づいて考案し，次に我々の研究室で研究・開発されてきたL.o.E.（Language-oriented Editorの略）[武田87][山田90]上で実現したので，ここに報告する．なお，実現にあたっては，PAX [Matsumoto 87]を利用した．

An Appropriate Syntax Error Analysing
based on Bottom-Up Chart Parsing
Hiroshi KATO, Masayuki TAKEDA, Kenzo INOUE
Science University of Tokyo

2．概念

ここでは，簡単な例に基づいて説明するとしよう．

次のような文法（文法1）が与えられたとする．

(1)	S → A B	(6)	C → c
(2)	B → C D E F	(7)	E → e
(3)	B → D	(8)	F → f
(4)	D → G H	(9)	G → g
(5)	A → a	(10)	H → h

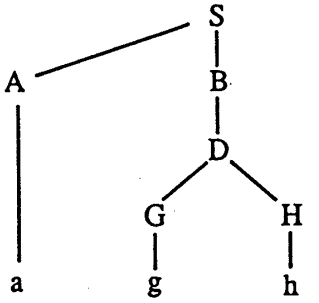
文法1

ここで，この文法をPascalの文法と対比させると，(4)が代入文に対する生成規則，(2)がfor文に対応する生成規則，終端記号cがfor，終端記号の並びe,fが，to n do .. を表しているのに注意して欲しい．

いま，文"a g h e f"について構文解析及び誤り処理を行なうとする．この文は，ちょうど1節におけるPascalの文に対応していることから，必ず何等かの誤り処理を行わなければならない．そこで，本手法を使って段階的に誤り処理を行う．

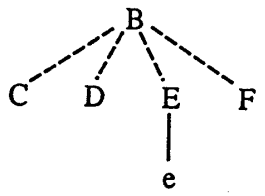
(1) まず，チャート法による構文解析を行なうと，次のような解析木を残して解析が失敗する（図1）．

図1



(2) ここで，解析が最も進んだ時点における字句，即ち，字句'e'に注目すると，'e'が構文解析部に受理されるためには，文法1の(2)と(7)から，図2のような部分解析木が生成されなければならない．

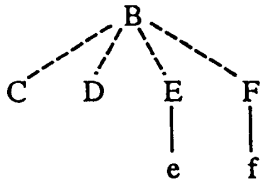
図2



(3) そこで，字句'e'以降の文字列がこのような部分解析木を生成し得るかどうか'e'以降を実際に解析する（図3）．この場合，解析に成功するので，'e'以降に

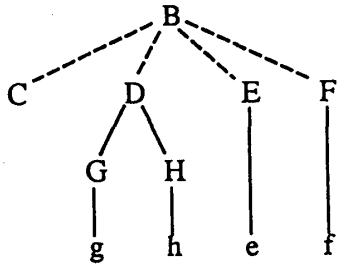
おいては図2のような解析木を生成し得ることがわかる．しかし，もし解析に失敗したなら，図2における見込は正しくないことになるので，他の生成規則に基づく見込をたてるか，それが不可能であれば，右隣の字句に基づいて見込をたててその字句以降を解析する．また，これを繰り返して，文の末尾まできてしまったら，(4) の処理をしないで (1) の結果からだけから誤りを診断する．

図3



(4) 図3 の部分木が (1) における解析時に生成されなかったのは，字句 'e' の以前に適切な字句がなかったからである．そこで，その適切な字句を探すべく，今まで解析してきた方向とは逆の方向で文を解析する．すると，その部分木を生成するには必要なのに，実際には存在してない字句が何であるかが明らかになってくる（図4）．

図4



(5) したがって，(1) ～ (4) の結果から次のような2つの誤り指摘が行なえる．

[1] 字句 'c' が存在していない．

[2] 字句 'e' 以降は余分な文字列である．

以上の結果から，1節で挙げた Pascal の誤った文でも適切に診断できる．

3．L.o.E. における実現

ここで，L.o.E. について簡単に紹介することにしよう．L.o.E. には次の2つの特長がある．

(A) 文法を知識として与えるので，文法さえ適切なものを取り替えればどんな手続型言語にも対応できる．

(B) L.o.E. にプログラムを入力すると，そのプログラムが構文・静的意味の2点から誤りがないか検証することができ，プログラムの編集・誤り検知を同一の環境で効率よく行なうことができる．静的な意味誤りに関しては，属性文法を利用した属性評価により適切な意味誤りが指摘できる．

したがって，L.o.E. に本手法を組み込むことは，誤り診断機能を強化することになるので，L.o.E. そのものの性能の向上に寄与することになる．

さて，L.o.E. に本手法を組み込むためには，チャート法を何等かのプログラム言語で実現されなければならない．そこで，我々は並列論理型言語 GHC で表わされ，チャート法と等価な解析手法である PAX を利用して実現することにした．GHC を使うことの利点としては，2節の (2) (3) の処理の過程で複数の見込に基づく解析を並列に実行できることが挙げられる．

具体的には，最初にトップダウン予測付解析を行なって，最も解析が進んだ時点での字句プロセスを発見し，その字句プロセスに2の (2) に対応するよう適切な識別子を与えて (3) に対応する解析を実行し，それに成功したら逆向きに解析するといった手順を踏む．

4．まとめ

本論文では，構文解析が失敗した点を中心に，部分的に構文解析をやりなおすことによって，適切な構文誤り箇所とその誤り原因を指摘できる誤り解析処理手法について述べた．これは，チャート法という構文解析アルゴリズムの枠組みに基づいているので，部分統制チャート法に基づく構文解析法 [Matsumoto 83][畝見 80] ならどれでも適用できる．

今後の課題としては，意味の誤り診断についても同様な手法を考案し，意味の面も含めて，より適切な誤り診断をできるようにすることが挙げられる．

また，チャート法そのものが CFG を受入れることから，対象言語を自然言語に拡大することによって，自然言語における文推敲システムを実現できるのではないかと考えられる．

参考文献

[Kay 80] M.Kay: Algorithm Schemata and Data Structures in Syntactic Processing, CSL-80-12, XEROX Palo Alto Research Centor, Oct., (1980)

[Matsumoto 83] Y.MATSUMOTO. et al.:BUP:A Bottom-Up Parser Embedded in Prolog, New Generation Computing, Vol.1, No.2, pp.145-158, (1983)

[Matsumoto 87] Y.MATSUMOTO: A Parallel Parsing System for Natural Language Analysis, New Generation Computing, No 5, pp. 63-78, (1987)

[畝見 80] 畝見達夫： 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)

[武田 87] 武田正之： 知識ベースに基づく Language-oriented Editor, 情報処理学会論文誌, Vol.28, No.11, (1987)

[山田 90] 山田雅彦他： 意味再解析を考慮した GHC における Language-oriented Editor の研究，情報処理学会 第 40 回全国大会 公演論文集, (1990)