ショートノート

大局的情報の利用を考慮した誤り回復構文解析の一手法†

大石優子†† 阿部圭一††

本論文では，誤り回復コンパイラに対する一つの提案を述べる．この提案は，語い解析の段階で式および代入文を一まとまりの単位として認識することによって，構文解析段階での先読みによる誤り回復能力を高めようとするものである．これによって，大部分の誤り回復は局所的な構文解析によって行うことができ，残された誤りも大局的構文解析によって回復できる．この方法は，現在実用に供されているコンパイラに比べて，誤りの出現により原プログラムが読みとばされる部分を少なくすることができる．

1. まえがき

コンパイラは，正しいプログラムを受け付けて翻訳するだけでなく，誤りのあるプログラムに対して適切なエラー・メッセージを出力しなければならない．このためには，誤りが発見されてコンパイラが正常な構文解析を進めることができなくなったとき，適当な誤りを仮定して誤り状態から抜け出す必要がある．これを誤り回復という[1),2)]．誤り回復は，プログラマが犯した誤りをコンパイラが勝手に訂正して翻訳すること(誤り訂正)を意図するものではなく，プログラムの残りの部分を解析するために最も可能性の高そうな誤りを仮定する操作である．

誤り回復の方法は二つに大別することができる．

(1) 誤りが発見される直前までの原プログラムは正しいと仮定し，それによって定められる適当な記号の集合の中のどれかの記号が現れるまで原プログラムを読みとばす（いわゆるパニック・モード）．

(2) 誤りが発見された点からある長さにわたって記号を先読みすることにより，その範囲内で最も可能性の高いと思われる誤りを仮定し，誤り回復を行う．

方法(1)は実用に供されているコンパイラでよく採用されている．この方法の欠点は，後述のように，Algol や Pascal 等，文のレベルで再帰的な構文規則をもつ言語においては，広い範囲にわたって原プログラムの読みとばしが起こりうることである．1回のコンパイルによってできるだけ多くの誤りを発見するという目標からは，このような広範囲の読みとばしは好ましくない．

方法(2)のなかで代表的なものは Graham らによる方法[3)]である．この方法においても，先読みする記号の個数を制限せざるをえないため，先読み部分に長い式などがあると先読みの効果がなくなり，上と同様の読みとばしを生ずる．

コンパイラの誤り回復における上記の問題点に対して，本論文では以下の提案を行う．

(1) 語い解析の段階で，式および代入文を一まとまりの単位として認識することにより，方法(2)の先読みの効果を高める．

(2) (1)の結果を用いて，構文解析段階での誤り回復の大部分を，式または代入文をはさんで隣り合う二つの記号の比較（以下ではこれを局所的解析と呼ぶ）だけで行う．

(3) 残された誤りの回復を大局的解析により行う．

2. 誤り回復構文解析のための構文情報

提案する誤り回復コンパイラでは，誤り回復構文解析を行うために，語い解析の段階で通常の記号の列のほかに構文情報の列を出力する．構文情報とは，記号列の中の一つの式および代入文を構成する部分をまとめて，それぞれ一つの記号 EX および AS で表したものである．式および代入文は，変数名，定数，演算子(代入文では代入演算子を含む)，および ( ) [ ]，等の記号が連続する部分であるから，語い解析の段階

† A Method of Utilizing Global Information in Error-Recovering Syntax Analysis by YUKO OISHI and KEIICHI ABE (Dept. of Information Science, Faculty of Engineering, Shizuoka University).

†† 静岡大学工学部情報工学科

〈ブロック〉 ::= **begin** 〈文〉 {; 〈文〉} **end**
〈文〉 ::= 〈単純文〉 | 〈構造文〉
〈単純文〉 ::= 〈変数〉 := 〈式〉 | 〈空〉
〈構造文〉 ::= 〈ブロック〉 | 〈**if** 文〉
〈**if** 文〉 ::= **if** 〈式〉 **then** 〈文〉 | **if** 〈式〉 **then** 〈文〉 **else** 〈文〉

図 1 仮定した文法
Fig. 1 A model grammar.

でも容易に検出することができる.

以下では,説明の便宜のために,**図1**に示す簡単な文法を仮定し,最終の認識目標(文法の初期記号)は〈ブロック〉であると考える.〈式〉以下の定義は省略してあるが,通常の定義,たとえば Pascal-S のそれを想定する.このとき,構文情報は次の記号からなる.

- **begin end**; **if then else**(これらを分離記号と総称する)
- AS(代入文)
- EX(式)

誤り回復は次の仮定のもとに行う.これらの仮定は制限的なものではない.なぜなら,これらの仮定が満たされない箇所に対しては,結果的に1章で述べた方法(1)が適用されることになるだけだからである.

(1) AS または EX を挟んで(あるいは挟まないで)相続く二つの分離記号を両端とする区間を局所的構文解析で一度に扱う対象とするとき,この区間内では構文情報レベルでの誤りはたかだか一つしかない(これ以外に式のレベルでの誤りを許すことは可能である).

(2) 語い解析において,式でも代入文でもない部分を誤って式または代入文と認識することはない.このような誤りはキーワードの綴り誤りによって生じ,それは変数の宣言等の意味情報を用いてほとんどの場合に訂正することができるから,この仮定は不合理なものではない.

3. 誤り回復構文解析

構文解析は,(1)式および代入文の解析,(2)局所的解析,(3)大局的解析の3段階に分けて行う.式および代入文の解析には従来の方法(SLR(1))を用い,式または代入文中の誤りの個数と訂正された式または代入文を得る.以下では,(2),(3)について説明する.

3.1 局所的解析

構文情報の列中で,一つの分離記号から次の分離記号までの間(両端の分離記号を含む)を局所的解析で一度に扱う対象とする.2章の仮定(1)より,この1区間の形は次の3種のいずれかに限られる.

表 1 分離記号と式,代入文との関係
Table 1 Compatibility of separating symbols with expressions and assignments.

α_1 \ α_2	;	**end**	**begin**	**if**	**then**	**else**
;	D, AS	D, AS	Λ	Λ	ϕ	ϕ
end	Λ	Λ	ϕ	ϕ	ϕ	Λ
begin	D, AS	D, AS	Λ	Λ	ϕ	ϕ
if	ϕ	ϕ	ϕ	ϕ	EX	ϕ
then	D, AS	D, AS	Λ	Λ	ϕ	D, AS
else	D, AS	D, AS	Λ	Λ	ϕ	D, AS

Λ: α_1 と α_2 の間には AS も EX も空文も挿入できない.
D: 空文を挿入できる.
ϕ: $\alpha_1\alpha_2$ という形も $\alpha_1 X \alpha_2$ という形も許されない.

(i) $\alpha_1\alpha_2$, (ii) $\alpha_1 X \alpha_2$, (iii) $\alpha_1 X_1 X_2 \alpha_2$

ここに,α_1, α_2 は分離記号,X, X_1, X_2 は式または代入文である.

(iii)の形は,X_1, X_2 の間に演算子または分離記号を挿入するか,または X_1, X_2 のいずれか一方を消去することにより,(ii)の形に還元して扱う.たとえば,

if EX EX **then**→**if** EX+EX **then**

(i),(ii)の形に対しては,分離記号 α_1, α_2 の間に来ることのできる要素の表(**表1**)と比較して1区間のパターンが正しいか否かを調べる.誤りを発見したときには,α_1 ((i)の場合),$\alpha_1 X$ ((ii)の場合)の後に来ることのできる分離記号を表の左から順に探して α_2 の前に挿入する.そのような分離記号が存在しないときは,α_2 または $X\alpha_2$ または $\alpha_1 X \alpha_2$ 全体を消去する.

以下に,局所的解析による誤り回復の例を示す.

(a) **if then**→**if** EX **then**
(適当な論理式 EX の脱落を仮定する)
(b) **if** $x := y$ **then**→**if** $x = y$ **then**
(構文情報 AS を EX に置換し,式の解析をやり直す)
(c) **begin** AS **if**→**begin** AS; **if**
(d) **else then**→**else**
(e) **if** AS **end**→空

3.2 大局的解析

局所的解析によって大部分の構文誤りは回復できるが,**begin** ～ **end** と **if** ～ **then** ～ **else** ～ がからみ合った誤りについては,局所的解析で扱う区間よりも広い範囲の情報を必要とする.たとえば,原プログラムの一部に

if ～ **then** AS; AS **end else** ～

という部分があった場合,これを

if ～ **then** **begin** AS; AS **end else** ～

の **begin** が脱落したものと見て，これを挿入して誤り回復を行うのが自然である．しかし，この部分の前に，対応する **else** と **end** がまだ現れていない **if** ～ **then** と **begin** とがある場合，すなわち，

if ～ **then** **begin** ～ ; **if** ～ **then** ～ **end else** ～

という場合には，矢印で示した **begin** と **end** の対応づけを行わなければならない．これは，局所的解析だけでは不可能である．

このために，大局的解析では，スタックを用いてシフト・還元型の構文解析を行う．**end** または ; の直後に **else** がある場合とない場合とでは異なる解析・誤り回復を行う必要があるため，局所的解析の時点で，直後に **else** をもつ **end** と ; は別の記号に置き換えてある．

大局的解析によって行われる誤り回復の例を示す．

(a) **if** EX **then** AS; AS; **else**
　　→**if** EX **then** AS; AS;

(b) **if** EX **then begin** AS; AS; **else**
　　→**if** EX **then begin** AS; AS **end else**

4. 評　価

図1に示した文法に対して，本方法を用いた誤り回復コンパイラを試作した．比較対象として，FACOM 230 Pascal 処理系および MELCOM-COSMO Pascal 8000 処理系を選んだ．これらの処理系では，**then** 節中の複合文を導く **begin** が脱落して

if ～ **then** ～ ; ～ **end else** ～

という形が現れると，後続の **else** 節の内容は完全に読みとばされてしまう．われわれのコンパイラでは **begin** の脱落を仮定するために，**else** 節の中も正常に解析される．

実用を目指した誤り回復コンパイラとして最も進んでいると考えられる Graham らのコンパイラ[3]との比較は，本コンパイラでは文法を極度に制限しているため，行うことができなかった．方法を比較するかぎりでは，われわれの方法のほうが簡単で（複雑な先読みやバックトラッキングを必要としない），同程度以上の誤り回復の効果が得られるはずである．

5. むすび

語い解析の段階で，式および代入文をまとめて認識した構文情報を出力することにより，広い範囲の情報を利用した誤り回復構文解析を容易に行うことのできる手法を提案した．

本研究では，宣言部，字下げおよび改行から得られる情報の利用は考えなかった．実用になりうる程度のコンパイル速度という点も今後の検討課題である．このように，本論文で述べた方法を実用化するには，いくつかの具体的問題を解決しなければならないが，誤り回復コンパイラに対する一つの有望な提案を行った．

参考文献

1) Aho, A. V. and Ullman, J. D.; *Principles of Compiler Design*, Chap. 11, Addison-Wesley, Reading (1977).
2) 稲垣康善，大山口通夫：構文解析からみた最近の計算機言語理論の動向，情報処理，Vol. 23, No. 1, pp. 53-61 (1982).
3) Graham, S. L., Haley, C. B. and Joy, W. N.: Practical LR error recovery, *SIGPLAN Notices*, Vol. 14, No. 8, pp. 168-175 (1975).

(昭和57年10月18日受付)
(昭和57年12月6日採録)