A-027

信頼性を導入した構文エラー処理

Syntax Error Handling Method with RIiability

冷水 匠 † 竹森 彬 † 木山 真人 † 芦原 評 †
Takumi Hiyamizu Akira Takemo Masato Kiyama Hyo Ashihara

1 はじめに

プログラミングをする上でエラー処理は重要な処理である．複数のエラーに対処するために，エラーの発見場所を出力したり，エラーの状態から回復するエラー処理が望ましい．

構文上のエラーはパーサによって発見できる．しかし，既存のパーサジェネレータから生成されるパーサは基本的にエラー処理の機能を持たない．このため，パーサにエラー処理の機能を追加したい場合は，ユーザによる特別な処理が必要となる．

例えば，著名なパーサジェネレータである yacc から生成されるパーサにエラー処理の機能を追加するには，文法にエラー規則を追加する．適切なエラー処理を行うにはユーザがこのエラー規則を適切な場所に書く必要がある．これはユーザに大きな負担をかける．

また，yacc[4] のエラー処理では，エラーの修正候補は出力できない．プログラムを修正するためには，エラーの場所だけでなく，修正候補まで出力できるほうが望ましい．

上述した特別な処理の削減，修正候補の出力には，パーサジェネレータがエラーの発見場所だけでなく、修正候補も出力できるエラー処理機能を持つパーサを生成できればよい．

また，プログラムには複数のエラーが含まれている可能性がある．このためパーサは複数のエラーに対応できるほうが望ましい．

本研究では，エラー距離を用いて信頼性の概念を導入し，複数のエラーを指摘しその修正候補を出力するエラー処理を提案する．このエラー処理を本研究室で開発したパーサジェネレータ prometeus に実装した．

2 エラー処理アルゴリズム

2.1 フレーズレベルエラー回復

フレーズレベルエラー回復は，エラーが発見された際にトークンの挿入，削除，置換などを行い，エラーからの回復を図る方法である．本研究では，エラーから回復する手法としてこの方法を用いる．

フレーズレベルエラー回復では，修正の後にエラーから回復できたかどうかを判断する必要がある．また回復の後にエラーを発見した場合は，新しいエラーか，誤った修正による後遺症的エラーかを判断しなければならない．

これらの判断にはエラー距離を用いる．エラー距離については次節で述べる．

† 熊本大学大学院自然科学研究科
Kumamoto Graduate School of Science and Technology

2.2 エラー距離

エラー距離は，修正を行った後にエラーを発見した場所から次のエラーを発見する場所，または入力の最後までの距離である．ここでの距離は，解析したトークンの数である．エラー距離と事前に設定した閾値より，エラーからの回復の成否を判断する．

エラー距離が設定した閾値を越えていれば，その修正は正しいと判断する．閾値を越えていなければ後遺症的なエラーとみなし，その修正は間違っていると判断する．

エラー距離により回復の成否を判断する際は閾値が基準となる．このため閾値を適切な値に設定する必要がある．閾値の設定によっては間違った判断を行う可能性がある．

この問題を解決するために，本研究ではエラー距離を用いて”信頼性”の概念を導入する．これにより閾値を用いずに修正の可否を判断する．これについては 3 章で述べる．

3 提案手法と実装

3.1 提案手法

本研究では，エラー距離を”修正の信頼度”とみなす手法を提案する．この手法では，パーサはエラーを発見すると解析を停止し，自分のコピーを作成する．コピーは別々の修正を行い解析を再開する．コピーは新しいエラーを発見するか，入力の最後に到達すると解析を停止し，この際のエラー距離を本体に返す．エラー距離を受け取ったパーサは各修正のエラー距離をその修正の信頼度とみなし，一番信頼度の高い修正をそのエラーの修正と判断し出力する．信頼度が同じ修正候補がある場合はその全てを出力する．

信頼度の導入により，閾値を用いずにエラー修正の成否を判断できるようになる．

以下に提案手法の処理の流れを述べる．

1. エラーを発見した際，解析を停止する
   本手法ではコピーを作成し，コピーで修正を行う．
2. 修正候補を選出する
   挿入や置換を行うトークンを選出する．今回は現在の状態からシフトや還元を行えるトークンを action 表を用いて選出する．
3. コピーを作成する
   エラーを発見する直前の状態のパーサのコピーを作成する．
4. 作成したコピーでそれぞれの修正を試行する
   修正の方法は次の 3 つである．
   - 削除
     入力に不必要なトークンがあったと仮定し，トークンを読み飛ばして解析を続ける．
   - 挿入

入力にトークンの不足があったと仮定し，トークンを挿入して解析を続ける．
- 置換
入力に間違ったトークンがあると仮定し，トークンを別のトークンに置換してから解析を続ける．

挿入，置換に用いるトークンは 2 で選出されたトークンである．

5. 解析が停止したときのエラー距離を本体に返す
コピーは解析を続けていき，解析が停止した時点でのエラー距離を修正の信頼度として本体に返す．
6. 本体が信頼度が一番高い修正を行い解析を続ける
信頼度が一番高い修正を行い，解析を続ける．
7. 解析が終了するとエラー情報を出力する
解析が終了すると，エラーを発見した場所，エラーの修正候補などのエラー情報を出力する．

3.2 prometeus

prometeus は Ruby で実装された LALR パーサジェネレータである．実装に当たっては参考文献 [2] に記された機構を参考としている．

prometeus は AdvencedParser クラスという拡張性を持たせるためのクラスを保持している．AdvancedParser クラスにはシフト，還元，エラーという LALR 構文解析の動作の前後に実行される四つのメソッドが用意されている．

機能追加を行う際には AdvancedParser クラスを継承し，機能追加を行いたい動作に対応するメソッドを上書きする．これによりユーザは機能追加を容易に行える．

3.3 エラー処理の実装

本研究では提案手法を 3.2 節で述べた prometeus に実装した．エラー処理は追加機能として作成し，prometeus 本体には変更を加えていない．このためユーザは文法を入力するだけで，今回提案するエラー処理の機能を持ったパーサの生成を行える．

ただし，prometeus では次のトークンを取得する処理はユーザ依存となっている．このため，エラー処理を行うにはいくつかの処理を追加する必要がある．必要となる処理を表 1 に示す．

表 1　追加処理

追加処理	内容
back_token	解析するトークンを一つ戻す
insert_token	トークンを挿入する
remove_token	トークンを削除する
replace_token	トークンを置換する
last?	解析が最後か判断する
line	現在解析中の行を返す
line=	行を書き換える

4 評価

4.1 評価方法

提案手法を実装した prometeus により生成されたパーサにエラーを含んだプログラムを入力する．その際，パーサがエラーを指摘できるか，正しい修正候補を出力できるかを確認する．

4.2 評価環境

エラーを含んだプログラムとして参考文献 [3] に示された Pascal で書かれた 13 個のプログラムを使用する．

4.3 評価結果

評価の結果，入力した 13 個のプログラムの内の 7 個のエラーを指摘し，修正候補を出力できた．7 個の内の 4 個が今回の目的である複数のエラーに対しての全ての修正候補を出力した．残りの 3 個は正しい修正候補を出力できたが，他にも修正できる候補があった．

エラーを指摘できなかった 6 個の文法は，フレーズレベルエラー処理では対応できない，グローバルなエラーを含んでいた．

5 まとめ

本研究では，エラー距離に信頼性の概念を導入したエラー処理の提案を行った．また，この手法を用いて複数のエラーを指摘し，その修正候補を出力するパーサを生成するパーサジェネレータの開発を行った．評価の結果，ローカルなエラーに対してはエラーを指摘し，修正候補を挙げられた．

今後の課題としてはグローバルなエラーへの対応が挙げられる．今回はフレーズレベルエラー処理によりローカルなエラーに対応している．このためグローバルなエラーには対応できない．今後，グローバルなエラーに対応できるエラー処理を開発する必要がある．

参考文献

[1] 佐竹力，中井央:オブジェクト指向に基づいた構文解析器生成法の提案，情報処理学会論文誌:プログラミング，Vol.45，No.SIG12(PRO23)，pp.25-38(2004)

[2] 真幡康徳，中井中央:構文解析器生成系と構文エラー処理，情報処理学会プログラミング研究会発表資料 (2005)

[3] Micheal Burk, Gerald A. Fisher Jr:A PARTICAL METHOD FOR SYNTACTIC ERROR DIAGNOSIS AND RECOVERY, ACM, (1982)

[4] Johonson, S.C.:YACC:Yet Another Compiler Compiler, UNIX Programmer's Manual, Vol.2, Holt, Rinehart, and Winston, New York, NY, USA, pp.353-387(1979). AT&T Bell Laboratories Technical Report July 31(1979)