

プログラム理解に基づいた知的なエラー検出

4 Q-3

若林 茂

(神戸市立工業高等専門学校)

1. はじめに

近年のコンピュータの普及にともない、現在、単に情報専門学科だけでなく、広くプログラミング教育が行われている。その際、学習者はコンピュータからのエラーメッセージを見ながらデバッグを行っている。しかし、このコンピュータからのエラーメッセージは初心者(プログラミング入門教育の学習者)には分かりづらいものが多い。たとえば、; (セミコロン) が1つ抜けていたり、余分に入っていたときに、ピントはずれのメッセージが出力されたり、構文上はエラーではないが学習者の意図とは異なる思いがけない動作をすることがある。(図1) それはコンピュータがプログラム言語の文法に基づいて解析を行い、不適当な部分をチェックしてエラーメッセージとして出力するからである。それに対して、人間の教師は学習者のプログラムを細かいミスにとらわれずに、改行や字下げ(インデントーション)の情報を積極的に使いながら全体から部分にトップダウンに調べ、学習者の意図を把握する。そして、その学習者に合った的確なアドバイスを与える。

本研究ではそのような人間の教師が行っている解析過程をコンピュータで実現することにより初心者にとって分かりやすいエラーメッセージの指摘を試みる。対象は本校電子工学科1年生の情報処理の授業である。演習プログラムはPascalで書かれた10数行から2~300行程度のものである。

2. プログラム理解の方針

2.1 PAD

プログラム構造を表現する図式として、問題分析図(PAD)がある。PADでは処理の流れを接続・選択・反復という3種類の箱で表現する。授業でも、プログラミングにとりかかる前に与えられた問題を解く手順をトップダウンに考え、この3種類の構造の組合せで表現するよう指導している。したがって、本システムでも学習者のプログラムを字下げの情報を使ってPADで表現することを考える。

2.2 プロダクションシステム

解析はプロダクションシステムで行う。学習者のプログラムを調べる際の教師の知識をルールで表現する。主なルールには次のようなものがある。

- ・ 接続について対応するbeginとendは同一行にあるか、複数行の場合は列位置が同じで各処理は字下げされている場合が多い
- ・ 選択(if文, case文等)、反復(for文, while文等)の処理対象は字下げする
- ・ 接続の各処理(同じレベルの処理)は列を揃えてかく

3. プログラム理解システム

3.1 システム概要

本システムは前処理部・ブロック化部・エラーアドバイス部から構成されている。学習者の作成したプログラムを前処理部で処理の最小単位(アトム)に分割し、それがいくつか集まって文となる。それをブロック化部で接続・選択・反復の組合せで構造化する。それを受けてエラーアドバイス部で適切なメッセージを出力する。本システムはprologで記述されている。

3.2 前処理部

前処理部では学習者のプログラムを空白、セミコロン、改行によって処理の最小単位(アトム)に分割する。

```
atm(Ano, ['文字列'], Lin, Col)
```

ここで、Anoはアトム番号、Linは行、Colは列を表す。

3.3 ブロック化部

ブロック化部では前処理部で作成されたアトムから字下げ情報をもとにプロダクションシステムにより、ブロック化を行う。各ブロックは次のように表現される。

```
blk(Bno, [seq], [Ano1, Ano2, .....], [E])
blk(Bno, [itr, while, <条件>], [Ano1, Ano2, .....], [E])
blk(Bno, [sel, if, <条件>],
    [[Ano11, Ano12, .....], [Ano21, Ano22, .....]], [E])
blk(Bno, [sen], [Ano1, Ano2.....], Lin, Col)
```

ここで、Bnoはブロック番号である。ブロックの種類はseq(接続)、itr(反復)、sel(選択)で表す。senは文を表す。Ano1, Ano2,はアトム番号である。これは処理が進むにつれb(Bno)に変化していく。Eはエラー情報である。

```
while r<>0 do; →エラーではなく無限ループとなる
begin
  p:=q      →'変数にエラーがあります'と表示される
  q:=r;
  r:=p mod q
end
```

図1. エラーメッセージ例

Program Understanding-based Error Detection

Shigeru WAKABAYASHI

Kobe City College of Technology

3.4 エラーアドバイス部

エラー情報は、スペルミス・必要な語句の欠如・余分な語句の挿入などに分類されている。これらはブロック化されるときに蓄積されていく。エラーアドバイス部では、その情報をメッセージの形で出力する。

4. 実行例

誤りを含むユークリッドの互除法のプログラムを処理した結果を図2に示す。7行目行末の余分な ; と 12行目の begin~endが抜けているのが指摘されている。

図3はブロック化した結果である。33個のアトムが作られている。字下げに基づいて構造化されているのがわかる。ここではエラー情報(err)は独立して表現している。また、文(sen)もブロック(blk)とは別に表現している。

5. まとめ

現在は、プロトタイプシステムがようやくできた段階なので、ブロック化部のルールなど今後改良していく必要がある。また、実際の現場で使用して評価していきたい。最後にいくつかの問題点とそれに対する考え方を述べる。

①字下げの付け方の規則と個人的な好みについて

字下げの付け方には個人的な好みがあり、一通りのやり方を押し付けるのは問題がある。しかし、初心者には標準的な書き方を示し、それにしたがってプログラムを書かせることは必要だと考えている。その上で各自が自分にあった書き方を作って行けば良いだろう。

②処理時間の問題

解析に時間がかかって学習者への応答が悪くなるのが考えられる。これについては現在はプロトタイプシステムの段階なので深くは考慮していないが、実際に現場で使う時点では大きな問題である。ルールのプレコンパイルなどを考えていきたい。

```

1  program gcd(input,output);
2  var a,b,p,q,r:integer;
3  begin
4    read(a,b);
5    writeln('a=',a,'b=',b);
6    if (a<0) or (b<0)
7      then write('ERROR');
8    else begin
9      p:=a; q:=b;
10     r:=p mod q;
11     while r<>0 do
12       p:=q; q:=r; r:=p mod q
13     writeln('gcd=',q,'lcm=',a*b div q)
14   end
15 end.
```

12行目 p:=q; の前に begin がありません。
 12行目 q の後に end がありません。
 7行目 余分な語が入っています。
 write('ERROR'); → write('ERROR')

図2. 実行結果

```

wm(atm(1,[program],1,1)).
wm(atm(2,['gcd(input,output);'],1,9)).
wm(atm(3,[var],2,1)).
wm(atm(4,['a,b,p,q,r:integer;'],2,5)).
wm(atm(5,[begin],3,1)).
wm(atm(6,['read(a,b);'],4,3)).
wm(atm(7,['writeln('a=',a,'b=',b);'],5,3)).
wm(atm(8,[if],6,3)).
wm(atm(9,['(a<0)'],6,6)).
wm(atm(10,[or],6,12)).
wm(atm(11,['(b<0)'],6,15)).
wm(atm(12,[then],7,5)).
wm(atm(13,['write('ERROR')'],7,10)).
wm(atm(14,[else],8,5)).
wm(atm(15,[begin],8,10)).
wm(atm(16,['p:=a;'],9,12)).
wm(atm(17,['q:=b;'],9,18)).
wm(atm(18,['r:=p'],10,12)).
wm(atm(19,[mod],10,17)).
wm(atm(20,['q;'],10,21)).
wm(atm(21,[while],11,12)).
wm(atm(22,['r<>0'],11,18)).
wm(atm(23,[do],11,23)).
wm(atm(24,['p:=q;'],12,14)).
wm(atm(25,['q:=r;'],12,20)).
wm(atm(26,['r:=p'],12,26)).
wm(atm(27,[mod],12,31)).
wm(atm(28,[q],12,35)).
wm(atm(29,['writeln('gcd=',q,'lcm=',a*b'],13,12)).
wm(atm(30,[div],13,40)).
wm(atm(31,['q'],13,44)).
wm(atm(32,[end],14,10)).
wm(atm(33,['end.'],15,1)).

wm(blk(5,[seq],[24,25,26,27,28])).
wm(blk(4,[itr,while,[22]],b(5))).
wm(blk(3,[seq],[16,17,18,19,20,b(4),29,30,31])).
wm(blk(2,[sel,if,[9,10,11]],[[13],b(3)]))).
wm(blk(1,[seq],[6,7,b(2)]))).
wm(err(5,24,comp,[begin])).
wm(err(5,28,comp,[end])).
wm(err(2,13,nlwd,['write('ERROR')'])).
wm(sen(1,[6],4,3)).
wm(sen(2,[7],5,3)).
wm(sen(3,[13],7,10)).
wm(sen(4,[16],9,12)).
wm(sen(5,[17],9,18)).
wm(sen(6,[18,19,20],10,12)).
wm(sen(7,[24],12,14)).
wm(sen(8,[25],12,20)).
wm(sen(9,[26,27,28],12,26)).
wm(sen(10,[29,30,31],13,12)).
```

図3. ブロック化結果

なお、本システムの作成は本校電子工学科5年田中至君、樋口竹利君による。

【参考文献】

- 1) 上野晴樹：知的プログラミング環境—プログラム理解を中心に—, 情報処理, Vol. 28, No. 10, 1987. 10