ソフトウェア工学 76－4
（１９９０．１２．４）

プログラム認識に基づくプログラム診断

海尻 賢二
信州大学工学部

概要

プログラミングを総合的にサポートするプログラミング環境は種々実現されているが、教育的側面に重点をおいたものはない。我々はプログラミング教育を重点的にサポートするものとして知的プログラミング教育環境（ＩＰＴＥ）の構築を目指している。ＩＰＴＥの主要な機能の一つにプログラミング演習のサポートがある。本稿では初心者のプログラミング演習において、課題に対するプログラムを認識し、適切に診断するシステム（**PascalRecognizer**）について述べる。**PascalRecognizer** は問題知識とプログラム知識に基づきトップダウン的にプログラムを認識し、その動的意味誤り（論理的誤り）を指摘する。

PROGRAM DIAGNOSIS BASED ON PROGRAM RECOGNITION

Kenji Kaijiri
Faculty of Engineering,Shinshu University

Abstract

Many programming environments were implemented, but those are not for programming education. We plan to construct the Intelligent Programming Tutoring Environment (IPTE), which supports educational aspects of programming. One of the most important functions of IPTE is the support for programming exercize. In this paper we describe about the **PascalRecognizer**, which recognizes the students programs using problem knowledge and programming knowledge and diagnoses dynamic semantics errors of the program.

1 まえがき

1.1 知的プログラミング教育環境 (IPTE)

Mentor、Gandalf、Synthesizer等に端を発するプログラミング環境はもはや実用の域に達し、いくつかのワークステーションやパーソナルコンピュータ上で商用のプログラミング環境が開発されている（例えばMacintosh上の各種言語システム、HPのSoftBench、アステックのSaber-C等）。しかしそれらは主にプログラミングの作業効率の改善を目的としてプログラミングの種々の側面をサポートしており、プログラミング教育という面からのサポートはほとんど行なっていない。プログラミング教育としては次の様な項目が考えられる。

- 言語教育
  プログラミング言語の構文、静的意味を教える。
- アルゴリズム教育
  標準的なアルゴリズムについて、その動作、効率、特徴等を教える。
- プログラム設計法教育
  設計パラダイム、効率化、プログラム書法等について教育する。
- ツール教育
  プログラミングにおいて必須となるプログラミングツール（エディタ、デバッガ、プロファイラ等）の使い方を教育する。
- プラグマティックスの教育
  プログラミング作業の各ステップで生じる種々のillegal responseに対してどう対処するか。例えばコンパイラの誤りメッセージに対してどう対処すればよいかを教育する。

上記の様な教育を行なうという観点から、プログラミング教育をサポートするプログラミング環境に対しては次のような要求がある。

1. 対象は初心者であるので、プログラミングの各作業においてその作業にのみ専念できることが望ましい。
2. 各作業において、利用者のレベルに即した診断・誤りメッセージを出す。
3. プログラムの完成よりも、そのプロセスの学習が重要であるので、利用法のチュータリング、誤りに対する的確な理由づけ、等が必要である。

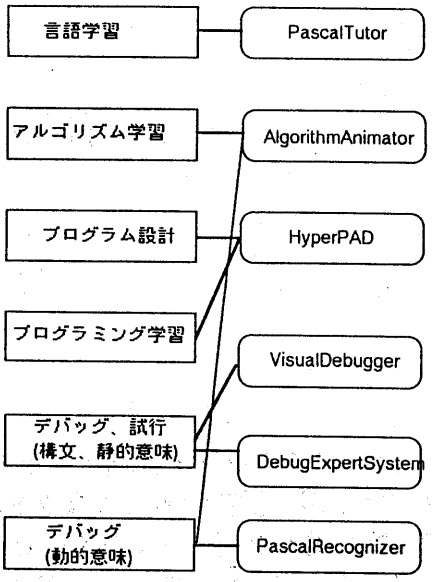

図 1: 知的プログラミング教育環境の構成

以上の観点から我々は学生のプログラミング教育全般を支援する環境として知的プログラミング教育環境 (IPTE) を考え、その実現を計画中である [10]。ＩＰＴＥにおいてはプログラミング教育のフェイズを図1のように捉え、各フェイズをサポートするものとして図1の右側のコンポーネントを試作中である。本稿ではこの中で言語学習と動的意味のデバッグをサポートするPascalRecognizer[11]について述べる。

1.2 プログラム認識

プログラミング演習はプログラミング教育において主要な要素である。PascalRecognizerはプログラミング演習における、プログラムの論理誤り（動的意味の誤り）の診断を目的とする。プログラミング演習は次のステップの繰り返しである。

1. 教師によるプログラミングの課題（問題）の提示。
2. 学生による問題に対する解答たるプログラムの作成。
3. 教師によるそのプログラムの正否の判断および指導。

プログラムの正否としては次の段階がある。

1) 構文誤りの有無

2) 静的誤りの有無

3) 動的誤りの有無

この内（1）、（2）はいわゆる解析によって認識できるものであるが、（3）はプログラム（プログラマ）の意図に依存するものであり、単なる解析だけでは認識できない。しかしプログラマ（特に初心者）がもっとも戸惑うのは（3）であり、この部分のサポートが是非とも必要になる。（3）が最も難しいのはこのレベルの誤りが単一の原因に基づいていない事による。即ち（1）、（2）は言語知識のみによって認識できるのに対して、（3）はアルゴリズムの知識、問題領域の知識などが必要となってくる。

（動的誤りの有無という意味での）プログラムの正否の判断法としては、

a) プログラムをブラックボックスとみて、標準解答プログラムと学生のプログラムに対するいくつかのサンプル入力に対する出力を比較することにより判断する [9]。

b) いくつかの標準解答プログラムを用意しておき、それと学生プログラムとの何らかの照合により判定する [2][5]。

c) 問題に対する解答の何らかの抽象的な記述を問題記述として与えておき、この問題記述に基づいて学生プログラムを（解析し、）認識することにより判断する [3][4]。

d) 一般的なプログラミングの知識に基づいて、学生プログラムの意味を推論することによりプログラム認識を行ない、判断する [6]。

等の方法がある。（a）は結果の正否しか判定できないので、誤っていた場合誤りの箇所等を識別できず、誤りの的確な診断は無理である。（b）は標準解答プログラムをベースとするため、新しい問題の登録は容易であるが、標準解答プログラムとの照合のとれるプログラムというのは限られる。ただ対象を LISP や Prolog に限定するならば、比較的その制御形式が限定されるため、この方法の適用も可能である [5]。（c）は問題の解をプログラミングにおいて通常使われる基本的要素（プランと呼ばれる事が多い）の何らかの組合せとして記述するもので、実現の上での種々の variation を一種の and-or 木で表現する。問題に対してこのような記述を与えることが（b）程は容易ではないが、基本要素を数多く用意する事により、多くの実現法、多くの誤りに対処することができる。（d）は（c）と同じようにプランをベースとするが、前もって与えたプランの組合せを認識するのではなく、知識ベースに登録されている個々のプランの認識（およびその合成）から動作を推論するものであるが、非常に領域を限れば可能であるが、一般的には領域が広くなり過ぎて無理である。

ソフトウェア工学的な立場からは reverse engineering のツールとして（d）の方式も使い得る。但しこの場合は完全に認識する必要はなく、reverse engineer の助けとなるような情報、即ちどのようなアルゴリズムに基づいて問題解決を図っているかに関する情報が抽出できればよい。

PascalRecognizer では教育用という観点から（c）のアプローチを採用する。プランとは”プログラムにおける特徴的な動作系列”とでもいうべきもので、プログラマがプランに基づいてプログラムを理解したり、作成したりすることは経験的にも明らかにされている [1]。

[3][4] ではこのプランをプログラムを理解するベースと考え、あるプラン集合の認識によりプログラムを認識しようと試みている。

本稿ではプログラマはプログラムをプランに基づいて段階的詳細化のパラダイムに載っとって設計している、との立場から、このプロセスをトップダウン的に再現するという形式でプログラム認識を行なう。

以下 2 章では PascalRecognizer の概要について、3 章では認識のベースとなるプログラム知識ベースの構成法について、4 章では認識法の概略を、そして 5 章では現時点での診断例についてそれぞれ述べる。

2 PascalRecognizer

PascalRecognizer は図 2に示す様に以下の 6 つの要素からなる。

1. Analysis module
2. Recognition module
3. Diagnosing module
4. 知識ベース
5. 学生モデル
6. UserInterface module

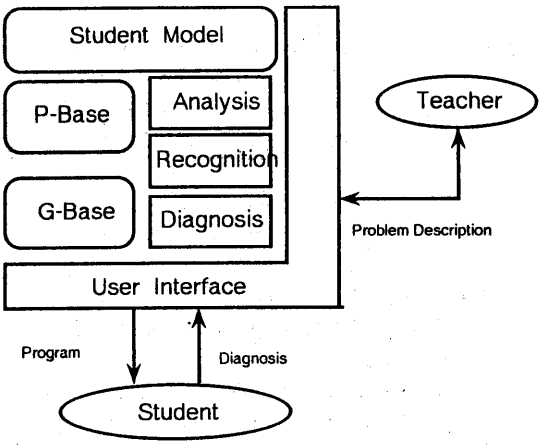

図 2: PascalRecognizer の構成

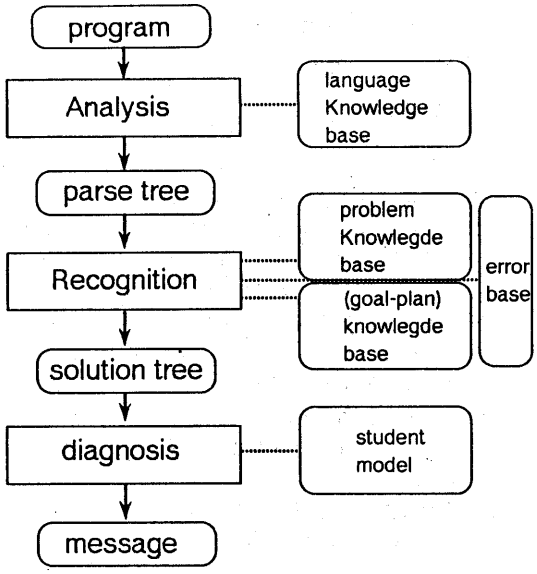

図 3: PascalRecognizer の処理フロー

知識ベースはプログラムの解析に関わる、言語（現在は Pascal）知識ベース、プログラムの認識に関わるプログラム知識ベース、そして診断に関わるバグ知識ベースの 3 つからなる。

PascalRecognizer はまず言語知識ベースに基づいてプログラムの解析を行ない、解析木に変換する。次にプログラム知識ベースに基づいて与えられた問題の解としてプログラムを認識し、解法木と呼ぶ中間表現を生成する。最後に診断モジュールはバグ知識ベースおよび学生モデルにもとづいて解法木の診断を行なう。この様子を図 3に示す。

PascalRecognizer は動的誤りの診断を目的としているので解析フェイズでの誤りは一切考慮しない（構文的、および静的意味の上からは正しいと仮定する。トータルなシステムとしてはこの様な誤りがあった場合は **DebugExpertSystem** を起動する）。以下では問題の記述と解析木をベースとして解法木を構成することを認識と、解法木からユーザの犯した誤りを推論することを診断と呼ぶ。

3 プログラム知識ベース

PascalRecognizer ではプログラム知識ベースを、プランをベースとして各問題のトップダウン設計法を記述した問題知識ベース（Ｐ－ｂａｓｅと呼ぶ）と、Pascal におけるプランの種々の実現方法を記述したゴール・プラン知識ベース（Ｇ－ｂａｓｅと呼ぶ）の 2 つから構成する。Ｐ－ｂａｓｅは問題固有の知識ベースであり、Ｇ－ｂａｓｅは問題とは独立なプログラミングノウハウに関する知識ベースである。

3.1 ゴール・プラン知識ベース

PascalRecognizer では（ゴール、プラン、テンプレート）という階層で、従来プランと呼ばれている基本的な動作系列の実現法を記述する。ここで

ゴール 基本的な動作系列に対して付けられた名称である。次の様なゴールを用意している。

- 番兵を除いた入力処理ループ
- 番兵を含んだ入力処理ループ
- ガード付き和
- ガード付き平均
- 代入
- 交換
- カウンタ型繰り返し、等

プログラマは言語の基本命令（for 文、while 文、if 文等）を組み合わせるのではなく、これらのゴールを組み合わせることによりプログラムを設計する。

プラン ゴールに対する種々の実現方法である。各実現方法はテンプレートとして記述する。例えばゴール”番兵を除いた入力処理ループ”に対しては次の様な 2 種類のプランが存在する。

- 入力そして while 文による繰り返し

- repeat 文による（入力・処理）の繰り返し

テンプレート プランに対する具体的な実現法である。上記2つのプランに対するテンプレートは次の様なものである。

- 入力そして while 文による繰り返し

```
read(@data);
while @condition do begin
  @process;
  read(@data)
end
```

- repeat 文による（入力・処理）の繰り返し

```
repeat
  read(@data);
  if @condition then
     @process
until not @condition
```

基本的にはプランに対してテンプレートは1つであるが、テンプレートに対して頻繁に使われるその変種または誤用があった場合、それらを正しいまたは誤った alternate として記述する。例えばゴール”ガード付き和”の1つのプラン（この場合は1つのプランしかないが）に対するテンプレートは次の2つがある。

- テンプレート1

```
if @data>0 then begin
   @sum:=@sum+@data;
   @number:=@number+1
end
```

- テンプレート2

```
if @data>0 then begin
   @number:=@number+1
   @sum:=@sum+@data
end
```

この例における2つの代入文は互いに独立であるので、その順序は無関係である。このような関係は問題知識ベースにおいても記述できるが、このようにテンプレートの相違としても記述できる。

この（ゴール、プラン、テンプレート）の階層関係を図4に示す。

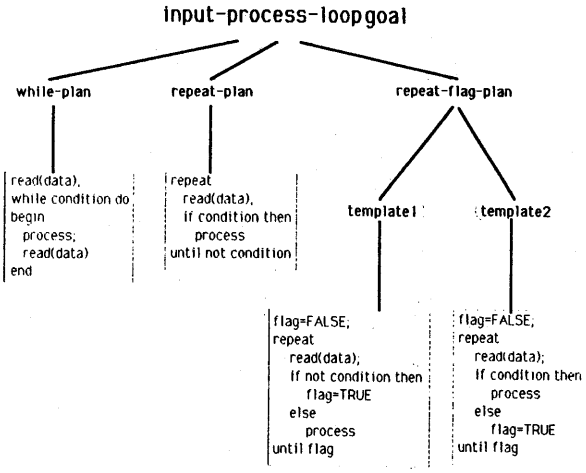

図 4: （プラン、ゴール、テンプレート）の例

3.2 問題知識ベース

問題知識ベースでは問題毎にゴールを基本命令とする段階的詳細化による設計を記述する。問題知識ベースは問題記述と設計記述の2つからなる。基本的に設計法の異なるものは異なった設計として記述し、その設計法を問題記述でリストアップする。

設計記述では初心者のプログラミング演習におけるプログラム診断という PascalRecognizer の特徴を考慮して、プログラムは基本的に次の構造の繰り返しにより構成されると考える（図5参照）。

```
前処理部------pre-part
何らかの制御--main-part
  処理--------in-part
後処理部------post-part
```

例えば次の様な問題を考える。

問題1 数の列を EOD が入力されるまで読み込み、それらの数の平均値を計算せよ。入力には負の数も含まれるものとし、平均には負の数及び EOD を含めないものとせよ。

上記の構造で考えればこの問題は次の様に設計される。

```
初期化と最初のデータの入力
EOD までの繰り返し
  加え合わせ
  データの入力
平均の計算
```

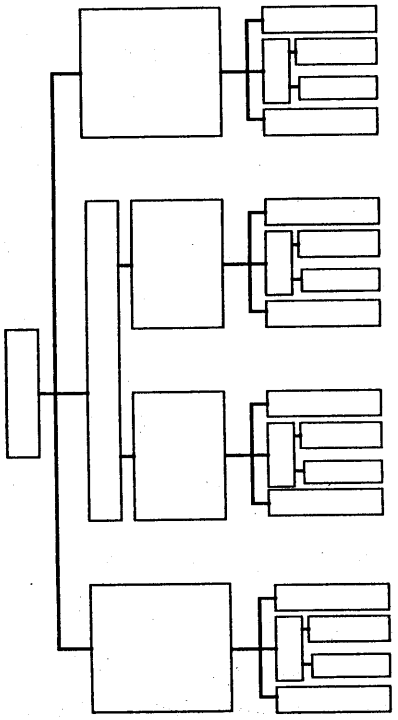

図 5: 階層的なプログラム設計

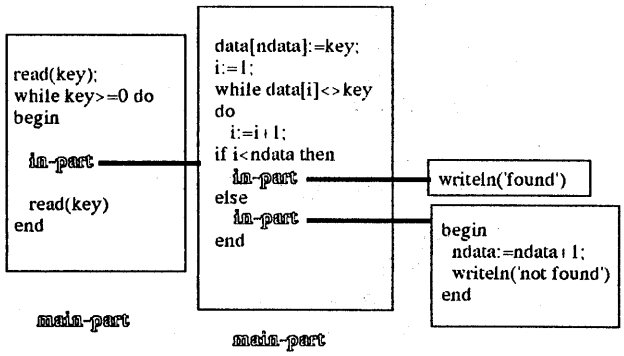

図 6: プログラムの階層構造の例

また次の問題を考える。

問題2 配列 data に ndata-1 個のデータが記憶されている。キー key を入力し、それが data に入っているかどうか判定せよ。入っていなければ ndata 番目のデータとして登録し、ndata を更新せよ。負のデータが入力されるまで繰り返し上記の処理を行なえ。配列 data の下限は 1 とする。キーが data に入っている時は "key is found" と、入っていない時には "key is not found" と出力する。

この問題であれば次の様に設計される。

番兵を除いた入力処理ループ
　番兵付き線型探索
　　メッセージ出力
　　処理とメッセージ出力

図 6 にこの設計に基づくプログラム例を示す。

4 認識

プログラムの認識における一番大きな問題は種々の variation の認識である。特に PascalRecognizer は認識による診断が目的であるから、論理的誤りのあるプログラムを認識する必要がある。但し一般的には正しいプログラム以外のすべてのプログラムが誤ったプログラムであり、それらをすべて認識して診断することは不可能である。そこで認識の範囲は概略のフローは異なっていないプログラムに限る。

この様な制限を設けてもまだその variation は数多く考えられ、その対応法が問題となる。PascalRecognizer では次のような階層的な variation の吸収法を利用している。

- 設計の相違
- 設計に対する誤ったゴール
- ゴールに対する誤ったプラン
- プランに対する誤ったテンプレート

ここまではデータとして与える。さらに以下のものはルールとして与える。

- テンプレートの標準的な variation
- 照合における variation

この詳細については別途報告する。

5 診断

PascalRecognizerでは解法木とバグ知識ベース（および学生モデル）にもとづいてプログラム診断を行なう。4章で述べた種々のvariation(error)に対しては個別に誤りメッセージを持っており、それらの変数名、キーワード等の実プログラムでの用語による置換により診断メッセージを生成する。

例。以下に問題1に対する学生プログラムに対するPascalRecognizeによる診断例を示す。

```
PascalTutorへようこそ プログラムファイル名を入力して下さい
プログラムは"DATA/SUM/1808.p"です
プログラムの解析を行ないます
プログラムの理解を行います
設計DESIGN1-1の検査
設計DESIGN1-2の検査
設計DESIGN1-3の検査
設計DESIGN1-3の検査
正しいプログラムとは認識されませんでしたので，誤り認識を始めます
設計DESIGN1-1の検査
設計DESIGN1-2の検査
設計DESIGN1-3の検査
設計DESIGN1-5の検査
認識結果
正しい設計（DESIGN1-1） 結果（0.405）
正しい設計（DESIGN1-2） 結果（0.0）
正しい設計（DESIGN1-3） 結果（0.0）
誤った設計（DESIGN1-5） 結果（0.0）
認識に成功しました
問題（PROBLEM1）に対するプログラム（"DATA/SUM/1808.p"）は
設計法（DESIGN1-1）を使っている
認識値は0.405です
しかし間違いが存在します

プログラムの診断を行ないます

001 program average(input, output);
002 var
003       n, s, a: real;
004       x: integer;
005     begin
006       read(n);
007       x := 1;
008       s := 0;
009       while n <> 99999 do begin
010         if n > 0 then begin
011         s := s + n;
012         x := x + 1
013       end
014       end;
015       a := s / x;
016       writeln(a)
017     end.

大域変数の利用状況
"平均値を代入する変数"として"a"を使っている
"番兵"として99999を使っている
        しかし問題記述では"EOD"を使っている
"個々の入力データを記憶する変数"として"n"を使っている
        しかし問題記述では"data"を使っている

"データの個数をカウントする変数（カウントには番兵を含めない）"として"x"を使っている
"データの和を代入する変数"として"s"を使っている

行番号（6 9-12）におけるgoal"番兵を除いた入力・処理の繰り返し"の実現において次の誤りを犯しています
  2つのread文の内の後半の繰り返し文中のread文を忘れている"

行番号（7）におけるgoal"xへの値の代入"の実現において次の誤り照合を行ないました
  定数0と1を等しいとみなした

行番号（15）におけるgoal"条件付きの平均"の実現において次の誤りを犯しています
  "xが0かどうかを検査するガードを忘れている"

行番号（15）におけるgoal"条件付きの平均"の実現において余分なコードが存在します

例終り
```

この診断では6行目の誤りと15行目のガード欠如の誤りを別個に指摘している。しかし見方をかえれば6行目でxの初期値を1としているために15行目に対してxが0かどうかのガードがないとも考えられる。

この様な各ゴールレベルの誤りに基づく推論により、より高位のレベルでの診断も必要ではあるが、

現時点では PascalRecognizer はその様な推論はサポートしていない。

6 むすび

（ゴール、プラン、テンプレート）の階層とトップダウン的なプログラム認識による初心者の Pascal プログラムの誤り診断システム PascalRecognizer について述べた。

現在実現しているプロトタイプでは認識速度は 5 章の例では数秒（kcl インタープリタ +sparc-1）であり、実際的な要求を満たしている。

本システムの目的はプログラムの初心者の演習問題で使われる様なトイプログラムの動的な（論理的な）誤りの認識による診断である。その意味からは非常に多様な誤りに対して診断を行なう必要がある。例えば問題 1 であれば次の様な誤りが観測された。

- 平均を最後に計算するのではなく、データを入力する毎に毎回計算する。
- バッチ処理的なイメージで毎回入力要求を出す。
- while 文を if 文とし、繰り返しを行なっていない。
- データを不必要に配列要素に読み込む。

この様な多様な誤りに対処するためには多くのバグ規則を用意する他、不完全な認識に基づく診断も考える必要がある。不完全な認識とは設計を構成するすべてのゴールの認識を仮定せず、一部のゴールの認識のみから設計を推論し、認識することである。

現在このような点も考慮して一問題当たり５０から９０個のサンプルについて約１０問題を対象として、９０％程度の認識率を目標にシステムを構築中である。

参考文献

[1] E.Soloway and K.Ehrlich:*Empirical Investigations of Programming Knowledge*,IEEE Transactions of S.E.,SE-10,5 (1984)

[2] A.Adam J.Laurent:LAURA,A System to Debug Student Programs, Aritificial Intelligence 15 (1980)

[3] W.Lewis Johnson,etc: PROUST:Knowledge-Based Program Understanding,ICSE (1984)

[4] 上野: アルゴリズム知識に基づくプログラム理解の枠組み, 人工知能学会研究会資料 (1989)

[5] William R.Murray:Automatic Program Debugging for Intelligent Tutoring Systems, Pitman, Readings (1988)

[6] M.T.Harandi and J.Q.Ning:*PAT:A Knowledge-based Program Analysis Tool*,Proc.Conf. Software Maintenance (1988)

[7] Dirk Ourston:Program Recognition,IEEE Expert winter 1989

[8] Rudolph E.Seviora:Knowledge-Based Program Debugging Systems,IEEE Software,May 1987

[9] 加藤他:*LISP-CAI* システムにおける診断と計画機能, コンピュータと教育研究会資料,11-2 (May 1990)

[10] 海尻: プログラム理解に基づくプログラミングチューター, 情報処理学会コンピュータと教育研究会資料 10-14 (April 1990)

[11] 海尻: プログラム理解に基づくプログラム診断システム, 情報処理学会 知識工学と人工知能研究会 68-6 (Jan. 1990)