プログラム理解に基づくプログラム診断システム *

5S－8

海尻 賢二†
信州大学‡

1 まえがき

プログラミング教育においてはプログラミング演習は不可欠であり、その適切な指導がプログラミング教育の大きな部分を占める。これを知的にサポートするためにはプログラムの理解および理解に基づく指導という機能が必要となる。我々は基本的にはPROUST[1]等で採用されているgoal-planの考えに基づき、段階的詳細化に基づきプログラム設計を辿るという形式でのプログラム理解を目的として、問題に対する解を（問題ー設計ーゴーループランーテンプレート）という階層で記述するという方式を提案し、この方式に基づき、初心者のpascalプログラムを診断するシステム（PascalTutor）を実現中である。

2 プログラム理解システム

PascalTutorはあらかじめ与えられた問題群に対して、（問題、プログラム）の対を入力として、その問題の解としてプログラムを理解し、診断を行なう。個々の問題に対してその種々の解法の表現を一種のand-or木として表現したものを問題解法木と、そして与えられたプログラムに基づいて問題解法木のorパスを特定したものを解法木と呼ぶ。プログラムはまづ解析モジュールで構文解析され、リスト形式の構文解析木に変換される。次に問題毎の問題知識ベース（問題解法木）とこの構文解析木とを照合し、プログラム理解を行ない、その結果としての解法木をつくる。次にこの解法木と教授知識に基づいてプログラム診断を行なう。問題知識ベースは問題固有の知識ベース（Pベース）と、問題独立の知識ベース（Gベース）からなる。

3 問題知識ベース

PascalTutorによるプログラム理解は問題解法木と学生プログラムとのパターン照合を基本とするが、この方式においては（1）「設計」を構成する各ゴール間の依存関係の表現、（2）一つの問題に対するプログラムのvariationの表現、の2つの問題がある。

（1）に対してはテンプレートとプログラムの照合の結果パターン変数にバインドされたプログラムリストを他のゴールとの照合に利用するという形式でゴール間の制約条件を表現する。このために垂直分割と水平分割という2種類の問題分割を採用している。

（2）に関しては問題の記述を（問題ー設計ーゴーループランーテンプレート）という階層構造で表現することにより、設計の誤り、ゴールの実現の誤り等を区別することを可能としている。さらに学生プログラムの標準化、テンプレートの標準／誤り変換そしてプログラムの等価規則によって問題に対する設計の数、ゴールに対するプランの数、そしてプランに対するテンプレートの数の軽減を計っている。

プログラム理解には認知心理学をベースとして人間によるプログラム理解をモデルとするアプローチもあるが、PascalTutorではソフトウェア工学的な見地からプログラム理解をとらえ、プログラムの段階的詳細化のステップを再現するという形でプログラム理解を行なう。そこで問題となるのは段階的詳細化の各ステップにおける基本要素である。あるレベルでは問題自体が基本要素になるし、また基本的な実行パターンも基本要素となる。

PascalTutorではこの基本的な実行パターンを中心にとらえ、これをゴールと呼ぶ。この様なゴールは抽象的な機能であり、これを実現する（pascalによる）プログラム（片）は種々存在する。これらの実現法をプランと呼ぶ。プランはゴールの特定の言語（PascalTutorではPascal）による実現法を記述するものであるが、その実現法にも種々の細かなvariationがあり得る。このvariation毎の実際の実現をテンプレートと呼ぶ。プランは一つ以上のテンプレートと対応する。

Pベースはある問題に対してどのような設計が存在するかという問題記述と、ゴール（および定義済みの問題）を利用して設計がどのように行なわれるかという設計記述の2つの記述からなる。

問題記述では種々の設計をリストアップする。問題の実現の種々のvariationの中には設計の間違いと考えるのが最も適当であるプログラムも存在する。そこで問題記述では正しい設計記述とともに誤った設計記述もリストアップする。

設計記述ではトップダウン的な問題の解法の分割（垂

---

*Program Diagnosis Based on Program Understanding
†Kenji Kaijiri
‡Shinshu University

```
(problem-frame
   'problem2
   "linear search and registration"
   '((variable "?data" "data")
     (variable "?count" "ndata-1")
     (variable "?key" "key"))
   '(design2-1 design2-2 design2-3)
   '(design2-4))

(design-frame
 'design2-1
 "linear search and registration"
 'problem2
 '((statement "?statement1")
   (statement "?statement2")
   (statement "?process"))
 '(goal1)
 '((main-goal (type goal) (name goal4)
              (apara "?key" "?key>=0" "?process"))
   (in-goal ((type goals)
             (object "?process")
             (goals
              (main-goal (type goal) (name goal1)
                         (apara "?data" "?count" "1" "?key"
                                "?statement1" "?statement2"))
              (in-goal ((type goal)
                        (object "?statement1")
                        (name goal2)
                        (apara "'key is found'"))
                       ((type goal)
                        (object "?statement2")
                        (name goal3)
                        (apara "?count:=?count+1" "'key is not found'"))))))))
```

図 1: 問題記述と設計記述の例

```
(goal-frame 'goal1
            'statement
            "linear search using sentinel\\
             set the sentinel at the last element \\
             and search until sentinel/key is found"
            '((variable "@data")
              (variable "@count")
              (constant "@init")
              (variable "@key")
              (statement "@statement1")
              (statement "@statement2"))
            '(plan1-1 plan1-2)
            '(plan1-3))


(plan-frame 'plan1-1
            "linear search using sentinel\\
             set the sentinel at the last element \\
             and search until sentinel/key is found\\
             You used while statement for this goal"
            'goal1
            '((variable "@i"))
            "@data[@count]:=@key; @i:=@init;
             while @data[@i]<>@key do
                @i:=@i+1;
             if @i < @count then
                @statement1
             else
                @statement2"
            nil)
```

図 2: ゴール記述とプラン記述の例

直分割）と順次的な問題の分割（水平分割）を組み合わせてあるパラダイムに添った設計を記述する。水平分割では主ゴール、内部ゴール、前置ゴール、後置ゴールの4つの形式のゴールを組み合わせて記述する。垂直分割ではパターン変数にバインドされたプログラム断片に対するパターンを（1）水平分割されたゴールの組合せ、（2）単一のゴール、（3）既定義の（別の）問題のどれかとして記述する。

Gベースは設計記述で使われるゴールとプログラムパターンたるテンプレートとの対応を記述するゴール記述と、テンプレートの実現法を pascal プログラムの形式で記述するプラン記述の2つからなる。

ゴール記述では正しいプランと誤ったプランを記述する。ゴール記述においても固有のパターン変数が使われるが、それをゴール変数と呼ぶ。ゴール自体及びゴール変数は指定された構文に対してのみ照合する。ゴール変数はいわばゴールの仮引数であり、設計記述でのゴールの引用時の (apara ....) の部分が対応する実引数になる。

プラン記述ではゴールに対するパターンを pascal の文（型）として与える。具体的に pascal の構文が使われるのはこの部分であるので、言語毎の「プラン記述」を用意することにより、複数の言語に対してプラン記述の切替えだけでプログラム理解を行なうことも可能である。

4 variation の取り扱い

PascalTutor では問題に対する種々のプログラムの variation を認識するために（1）（問題ー設計、ゴーループラン、プランーテンプレート）の3つの階層に分けた問題に依存した variation の記述、（2）標準的な variation および誤りのためのテンプレートの自動生成、（3）不完全な照合も許すパターン照合の3つのアプローチを採用している。これにより一つのプログラムに対して種々の認識が可能となり得るが、各設計に対して互いに素な1つ以上のキーとなるゴールを対応づけ、これらと学生プログラムをまづ照合することにより設計を特定するというアプローチにより解決している。

5 むすび

標準的な変換および誤り変換により問題に対する知識ベースをむやみと大きくすることなく種々のプログラム variation に対応することを可能としている。プログラム理解はあらかじめ定義された設計法にもとづいたプログラムしか認識できないという欠点はあるが、標準的な問題の設計法を教えるという観点からは問題ないと考える。

参考文献

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