ＷＥＢを利用したプログラムバグ診断システム

Program Diagnosis System using World Wide Web

海尻 賢二

Kenji Kaijiri

信州大学工学部

Faculty of Engineering, Shinshu University

＜**あらまし**＞プログラミング演習の補助を目的として、web を利用したプログラム診断システムを試作した。本システムは、１）問題自体も html ファイル形式で記述するために問題の準備が容易である、２）プログラム全体ではなく、穴開けにより特定の部分のみ入力させる方式を採用した。これにより、複雑な変種を避けることができ、診断能力も向上した、３）学生は web のブラウザを利用してシステムにアクセスするので環境を選ばない、といった特徴を持っている。

＜**キーワード**＞WWW、プログラミング認識、プログラミング演習、遠隔教育

1　はじめに

プログラミング言語及びプログラミングの学習においてプログラミング演習は必須の項目である。そしてプログラミング演習のサポートのためにはプログラム認識が必須となる。これまで教育を目的として種々のプログラム認識、診断システムが作られているが、基本的にプログラム全体を診断の対象としたものである。そのため

- 学生による自由なプログラム構成が可能なため、予想できない変種が出てきて認識しきれない
- 良くわかっていない学生の場合、解とはいえないような解を出してくる。この様な場合は標準的に用意したプログラムパターンとの照合という観点からは認識の範囲を越えてしまう。

のような理由から認識に限界が生じて、上記の意味での適切な診断が十分には行なわれていない。また問題の記述が複雑となり、簡単に大量の問題を用意すると言う事が難しくなっている。

そこでこれまでのプログラム認識の手法を参考にして以下の様な特徴を持つ教育目的のプログラム認識、診断システムを試作した。

- 穴空きプログラムの穴埋め問題としてプログラミング演習を捉える。これにより
  - 特定の部分のみの理解に限って診断できる。理解の範囲が限定でき、診断も特化できる。
  - 解としてのプログラムの大枠を指定し、着目したい点に重点をおいて問題を作ることができる（例えば関数の使い方）。最終的には全体としてのプログラムが作成できないと意味がないが、教育過程においてはこの様な方式も効果があると考えた。但し穴としてはプログラム中の任意の構文単位（変数、式、文、文列等）を可能とすることにより問題の単純化を防いでいる。
  - 変数名を指定できる。幾つもの変数を使う場合、変数名の variation が全体の大きな variation に継る。これを限定する事により単純化がはかれる。
  - 枠をはめていることにより学生にとってもヒントとなり、とんでもない解は減少する

  等の利点が得られる。
- システムの枠組としては遠隔教育も考慮して web を利用した。これによりユーザ側のプラットフォームの独立性が保たれ、どこでも利用できる様になる。またＧＵＩとしてｗｅｂブラウザを利用する事により、画面設計が容易になった。

- 問題文は解も一緒にして構成し、変換プログラムにより、ｗｅｂ用の問題ファイル（html ファイル）と、その後の診断ファイルに分ける。これによりコースウェア的に問題群を作っていくことも容易になった。

- バグ認識は標準解として与えられたプログラム断片と学生の与えたプログラム断片との解析木レベルでの照合により行なう（文献 [1]）。但し単なる照合では標準的な（許されるべき）variation も照合できない場合が多々あるので、標準化により対応する。また前節で述べたように標準解としてはプログラム断片以外にも文献 [9] で使っているプラン表現を利用する事により、種々の variation に対する対応を行なう事も予定している。

以下２章ではプログラム認識の方法論について、３章ではシステムの概要について、４章では認識、診断方法について、そして５章では評価を行なう。

2　プログラム認識

我々はリバースエンジニアリングの立場からのプログラム認識と、プログラミング演習の補助としてのプログラム認識を明確に分離する。前者はソフトウェア、プログラムアーキテクチュア、更には design decision の認識が目的であるが、後者ではプログラムの（意味的な）バグの認識を目的としている。本システムではプログラミング演習のコースウェアに組み込まれた演習問題における診断という立場から問題を捉える。

プログラムのバグ診断を目的としたプログラム認識の研究は多く報告されている [1, 2, 3, 4, 7, 9]。それらにおいて問題とされていることはプログラムの種々の変種をどの様に扱うかという事である。特に初心者のプログラムの診断の場合、予想もしないような回答が生じ、全体としての正誤以外は診断できない状況がしばしばが生じている。そこでまず変種の扱い方として何らかの形態での標準化、そしてプランという形態での一般化が考えられている。

しかしプランという形態の場合、プランの構造体として問題の解を記述する必要があり、プログラミング演習において多くの問題を用意するという立場からは容易ではない。また標準化の場合には良形なプログラムの場合はいいが、悪形プログラムの場合、適用が難しいという問題がある。プログラムの認識法としては標準プログラムとの照合による方法、プラン表現での正答（誤答）との照合による方法、標準化されたフロー表現での照合による方法等があるが、どれもプログラム全体を認識の単位としている。

本システムでは穴埋めという形式で認識の適用範囲を限定する事によりこの問題の解決を計っている。

3　プログラム認識システム

システムの概要を図 1 に示す。

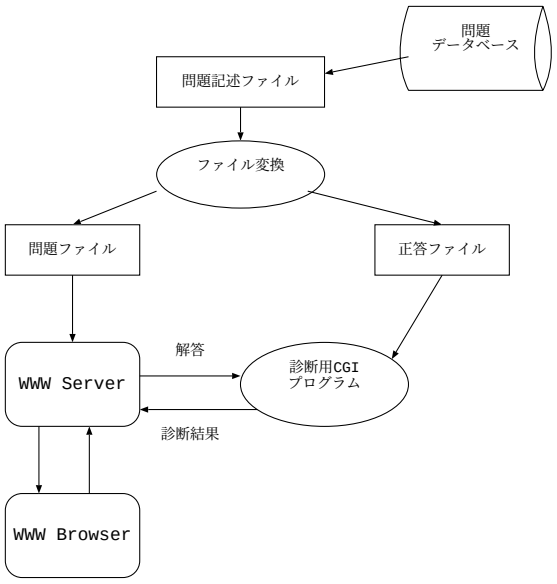

図 1: プログラム認識、診断システム

図 1 に示すように問題はデータベースに課題毎に格納されている。課題の記述は基本的には html 形式であるが、穴埋め部分に対する解答も合わせて記述するため、その部分の拡張を行なっている。

処理の流れを図 2 に示す。

課題全体は html 形式で記述されている。学生はその中から課題を選択する。各課題は穴に対して解答を含んだ形式で記述されており、学生の指定に従って課題表示のための html ファイルと穴毎の解答例に分割される。html ファイルはそのまま問題としてブラウザに送られ、学生に対する問題となる。学生は各穴に対してプログラム断片を入力する。そして診断ボタンを押す。するとサーバ側では照合診断のための cgi プログラムが起動し、穴に

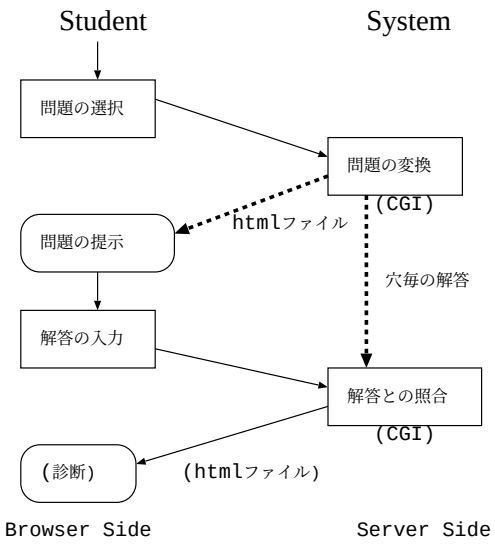

図 2: 問題の提示と診断の流れ

入力された解答と教師があらかじめ与えておいた標準解との照合が行なわれる。

照合においてはできるだけいろいろな解答の処理ができるように以下の処理を行なう：

- 解答の標準形式への変換
- 幾つかの解答を前もって与える
- 正答に対する標準的な誤答変換

従来この様な変種に対する対応としては上記の処理が行なわれてきたが、プログラム全体を対象とするために変種が極めて多くなり対応できないものが多くでた。本方式では概略のアルゴリズムなり、変数名なりを予め与えておくことが可能であるため、そのような変種はかなり少なくなる。また従来の方式では到底プログラムと言えないようなものまで診断の対象としなければならなかったが、本方式では枠組を与えておくために飛んでもない解答は出てこない。また各穴毎の診断を基本とするために範囲が限定され、予想もしないような二次誤りの可能性はほとんどなくなる。プログラム全体を対象とする場合、二次誤りにより診断精度が損なわれるが、適用範囲を限定することにより、二次誤りを減らし、従来の手法の適用性を高めている。

システムは解析部分を除き、言語依存性はない。現在はC言語を対象としている。システムはCと Perlを利用して実現している。画面の設計等の観点からは VBscript の採用も考えられるが、ブラウザが限定されるので採用していない。

4 問題の記述

問題は各課題を単位として、1）課題の記述、2）穴を含むプログラム本体、3）穴に対する標準解答集合、4）典型的な誤答群と対応する診断メッセージ、の4つの情報を一括して記述する。

問題の記述における穴埋めすべき問題部分は以下の形式で記述する。

```
＜問題記述＞ ::=＜問題名＞@@＜問題の解説＞@@
                ＜標準プログラム＞
＜標準プログラム＞::=/*ホール記述を含むプログラム*/
＜ホール記述＞::=＃＜正答のリスト＞
                ＃＜誤答のリスト＞＃
＜正答のリスト＞::=＜正答＞｜
                ＜正答＞＜正答のリスト＞
＜正答＞::=@＜正答指標＞@＜プログラムパターン＞
＜誤答のリスト＞::=＜空＞｜＜誤答＞｜
                ＜誤答＞＜誤答のリスト＞
＜誤答＞::=@＜プログラムパターン＞@
                ＜誤答メッセージ＞
```

問題に対する解答であるプログラムは、＜ホール記述＞を含むプログラムとして記述する。＜ホール記述＞は文法的には式又は文として認識されるものとして文法を拡張している。ホールとして記述された部分は html においては form または textarea によって実現するが、そのサイズは正答のサイズにしたがって決める。＜正答指標＞は問題全体で正答を識別するためのものであり、正の値の場合は同じ番号を持つ正答の組みが真の正答となる。すなわち2つ以上の穴の間での正答の関連づけに利用する。＜誤答メッセージ＞は対応する誤答が認識された場合の診断メッセージである。＜誤答メッセージ＞はオプションであり、ない場合には間違っている旨の診断のみが行なわれる。

＜プログラムパターン＞は認識すべき標準プログラム断片であり、式もしくは文としてのプログラム表現であるが、識別子の任意性を与えるために識別子に対応するパターン変数を用意している。パターン変数は＄と英字1字で表現する。2つ以上の別のホール記述で同じパターン変数が使われた場合には、それに対応する変数名はユニークである必要がある。1箇所でしか使われてい

ない場合には任意の変数名で良い。

実際の問題の記述例を図 3 に示す：

```
問題１
@@
標準入力から５文字以内の文字列を読み込み、その文字列を標準出力に出力するプログラ
ムを配列を一つ使って実現せよ。
@@
#include <stdio.h>
main()
{
  int i;
  #@0@char $a[6];#char $a[5];@ 文字列の最後にヌル文字が追加されることに注意
  @char *$a;@ ポインタではなく配列を使用して下さい#
  scanf("%s", #@0@ $a #&$a@ 配列の場合は配列名がそのままポインタになるので＆は不要#);
  #@0@
  i=0;
  do
  { printf("%c", $a[i]);
    i++;
  }while($a[i] != '\0');
  @0@
   i=0;
   do
   {printf("%c", $a[i]);
   }while($a[i++] != '\0');
  @0@
   printf("%s", $a);
  ##
}
```

図 3: 問題の記述例

この例においては３つの穴を与えており、それぞれに対して正答、誤答を与えている。この記述によって作られる問題画面を図 4 に、また診断の例を図 5 に示す。

5　認識、診断方式

診断は基本的には穴毎に行なうが、同じ番号（１以上）をもった正答がある場合にはそれらがすべて照合できて始めて正答とみなす。診断のための照合の手順を図 6 に示す。

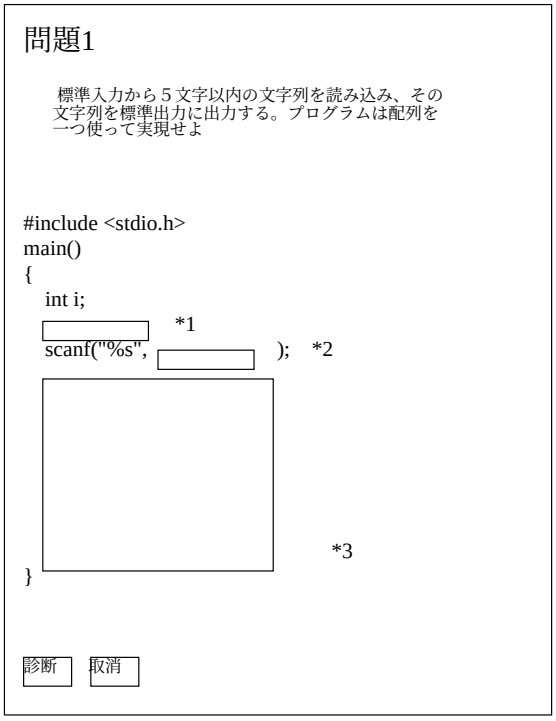

図 4: 問題画面の例

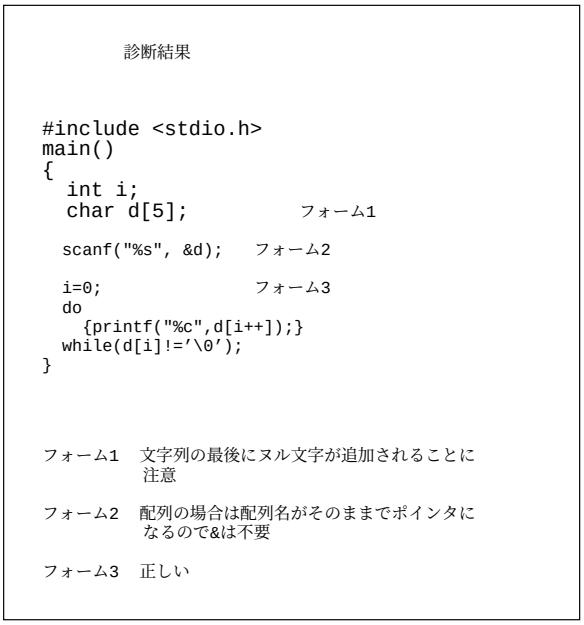

図 5: 診断画面の例

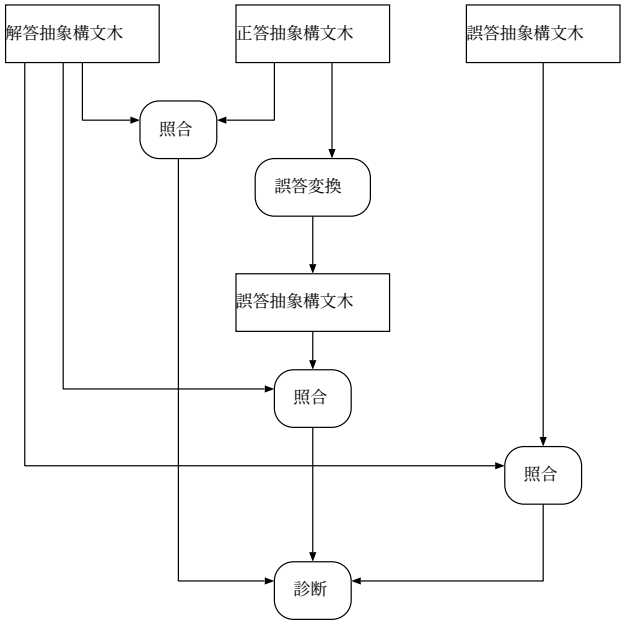

図 6: 診断の手順

5.1 照合

照合は基本的には図 7 の様な手順で記述される木照合である。

```
照合（解答、正答）
  ｉｆ 解答のラベル＝＝正答のラベル ｔｈｅｎ
    各子供について照合を行ない、すべて成功であれば
    照合成功
  ｅｌｓｅ
    照合失敗
  ｅｎｄｉｆ
```

図 7: プログラム照合手順

但しこの方法では交換律を満たす演算子の様な単純な変種の場合でも照合不可の場合があるので、以下の様な融通性を持たせている：

- 交換律を満たす演算子の場合、左右の子供を交換した場合の照合も試みる。
- 比較等順序のある演算子については一方に標準化したうえで照合する。

5.2 誤答変換

穴埋め形式であっても種々の変種が生じる。照合ができないものは単に誤りとして診断するが、できるだけ何らかのメッセージをつけて診断するための一般的な誤答変換規則を用意しておき、正答から誤答を自動的に作りだし、照合を試みる。一般的な変換規則としては以下の様なものがある

1. 等値判定の関係演算子と代入演算子の混乱
   正答 (a==0) に対する誤答 (a=0) であり、メッセージ（等値判定の関係演算子と代入演算子を間違って使っています）を添える。
2. 配列を宣言する時のサイズの間違い
3. ポインタ演算子の取り扱いに関する誤り
   &か ∗ をつけるべき場所についていない。

これらにより問題作成者の負担も軽減させる事ができる。

6 考察

6.1 問題記述について

問題は正答、誤答を一緒に記述する訳であるが、記号の区別が複雑で間違い易い。システムへの入力としてはこのままでもよいがＵＩという観点からは改良が望ましい。現在 GUI 形式の問題記述用のエディタを実現中である。

6.2 認識、診断について

認識は解析木の照合、診断は添付のメッセージを基本として行なった。これにより従来のプランを用いて全体を認識する方法 [9] に較べて認識部分の大幅な簡単化が計られた。しかし以下の様な問題点も明らかとなった。

- プログラム全体に渡る診断メッセージの必要性。
  各フォーム毎の誤答に対する誤りメッセージは用意できるが、組合せエラーに対する診断メッセージを別個に用意する事はできない。
- 多くの variation をまとめて表現するためのプラン記述の導入の必要性。

例えば同じ穴を２箇所で使っている場合、これの同じであるとの指示、また正答等のプランによる記述が必要である。しかしこれには問題記述の複雑化とのトレードオフの問題がある。

6.3 WEBの利用について

インフラとしてのWWWの利用は単純な、システムの構築という面からは有用であった。更に今回の様に多数の学生を対象とするシステム、又遠隔教育という面からも更に押し進めていくべきアプローチであると考える[10]。しかしながら以下のような問題点も明らかとなった。

- ユーザ側からの入力の手段としてはフォームを利用するしかないが、問題画面の設計という観点からは自由度がなく、十分なものとはいえない。
- システム側として認識のフロントエンドはcgiプログラムによる学生の解答の取り込みになるが、通常の利用ではstate-lessであるので、ITSの様に学生モデルなどを利用する場合にはさらなる考察が必要となる[8]。

現在サーバの高度化、cgi技術の改良等行なわれており、上記の問題は大きな問題とは言えない。むしろこの様なシステムの実利用を進めて、教育効果の面からの評価を行なう事の方が重要であろう。

7 結び

統合プログラミング演習システムの一貫として実現したプログラムのバグ診断機能を持つプログラミング演習システムについて述べた。本システムはプロトタイプが完成した程度であるので十分な評価はまだできていない。今後WebCationの利用が当学科でも予定されているので、それまでには完成させて評価を行ないたい。

プログラミングという名前で総称される科目に対して修得すべき数多くの項目を含んでいる。これらをすべてマスターして始めてきちんとしたプログラムが設計、実現できる訳ではあるが、それらを一括して教えようとするところに、初心者がプログラミングを難しいものと考える大きな原因があると考える。できるだけ個々の項目を分離してサポートできるツールの実現を同様なインフラの元で考えていきたい。

参考文献

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

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

[3] Charles Rich and Richard C. Walters: *The Programmer's Appentice*, Readings, ACM Press (1990)

[4] 藤井、渡辺、田中、杉江：*Pascal***プログラム教授システムにおける誤り同定法**、情報処理学会論文誌, 34, 3 (Mar. 1993)

[5] 何、池田、斉藤、溝口：**プログラミング教育のためのプログラム理解知識の構造について**、CAI学会誌, 10, 1 (March 1993)

[6] 海尻、吉田: **統計的手法によるデバッグ支援システムの構築**, CAI学会, Vol.11, No.2 (July 1994)

[7] 岡本、松田、安田：**アルゴリズム診断機能を有するCプログラミング学習支援システムの研究**, CAI学会誌, 11, 2 (June 1994)

[8] K.Nakabayashi, etl: *A Distributed Intelligent-CAI System on the World-Wide Web*, ICCE'95

[9] 海尻: **ゴール／プランに基づく初心者プログラムの認識システム**, 電子情報通信学会誌, Vol.J78-D-II, No.2 (Feb. 1995)

[10] Murray W. Goldberg: *World Wide Web - Course Tool: An Environment for Building WWW-Based Courses*, 5th International WWW Conference (May 1996)

[11] 関本、海尻: **プラン認識を利用したプログラミングスタイルの診断**, KBSE (July 1997)

[12] 服部、石井: **プログラミング演習の評価サポートシステムの構築**, 教育システム情報学会誌, Vo.14, No.1 (Apr. 1997)