プログラミング学習支援システムにおける出題意図に基づいた正誤判定機能の検討

A Method for Evaluating Students' Programs based on Intent of Questions in Programming Learning Support Systems

塩見 健太[*1], 朝倉 宏一[*2]
Kenta SHIOMI[*1], Koichi ASAKURA[*2]
[*1] 大同大学大学院情報学研究科
[*1] Graduate School of Informatics, Daido University
[*2] 大同大学
[*2] Daido University
Email: k.showme0310@gmail.com

あらまし：本稿では、プログラミング学習支援システムについて学習者の解答プログラムが問題の出題意図に沿っていない場合でも正解になってしまう可能性について焦点をあて、解決を目指す。教授者が用意した正解プログラムを分析し、出題意図候補を抽出する。教授者が候補より出題意図を設定することで、学習者の解答プログラムを出題意図に基づいて正誤判定することが可能となる。

キーワード：プログラミング学習支援システム，オンラインジャッジ，出題意図

1. はじめに

近年、プログラミングへの注目が高まりつつあり、プログラミングを学習することができるゲームや知育玩具など、プログラミングに関する様々な商品が発売されている。その理由の一つとして、学習指導要領にプログラミングが追加され、今では小学校からプログラミング学習が行われていることが挙げられる(1)。

プログラミング学習には様々な方法が考えられるが、本研究ではプログラミング学習支援システムを活用したプログラミング学習に焦点をあてる。プログラミング学習支援システムは、オンラインジャッジとも呼ばれ、プログラミング開発環境を構築することなく Web 上でプログラムを作成する機能や、学習者が作成したプログラムの自動採点機能など、学習者にとって便利な機能が搭載されている。

一般的に、プログラミング学習支援システムの自動採点機能は、入力とそれに対する正しい出力が対となったテストケースを用いて判定する。すなわち、プログラムの中身を確認することなく、プログラムの正誤を判定するブラックボックステストになっている。そのため、問題の出題意図や制限事項に沿っていないプログラムでも、入力に対する出力が正しければ、正しいプログラムであると判定されてしまい、プログラミングの学習という観点からは問題である。例えば、C 言語の学習において、for 文を用いた複数行表示の問題を出題したとき、for 文を使用せず printf()等を複数行記述したプログラムでも正解となる可能性がある。また、for 文を使用した例題プログラムと同じ動作をするプログラムを、while 文を用いて作成する問題では、例題プログラムそのままであっても正解と判断されてしまう。

上記の問題に対して、モデル検査を用いて評価する手法(2)が報告されているが、検査項目として有界モデル検査器である CBMC のアサーションを用意しなければならず、労力がかかってしまうのが問題である。そこで、本研究では教授者の出題意図に基づいたプログラムの正誤判定手法を提案する。上記の例では、「for 文を使用する」や「while 文を使用する」が出題意図にあたる。教授者が用意した正解プログラムを分析することで出題意図を抽出し、それを学習者の解答プログラムの正誤判定に利用する方法について述べる。

2. 提案手法

教授者の出題意図を把握し出題意図に基づいて学習者の解答プログラムを判定するため、以下の手順を踏む。

1. 教授者が正解プログラムを登録する
2. 正解プログラムを解析し、出題意図候補を抽出する
3. 教授者が出題意図候補から問題にあった出題意図を選択する

2.1 出題意図候補抽出手法

教授者が登録した正解プログラムを、LLVM(3)を使用して解析し、抽象構文木を出力する。出力された抽象構文木を分析して出題意図となる可能性がある要素を抽出する。抽出される要素は「関数 (Functions)」「変数 (Variables)」「ループ (Loops)」「条件分岐 (Branches)」「関数呼出し (Calls)」の 5 種類であり、プログラム中の出現箇所（行数）とともに抽出される。例として、図 1 のプログラムに対して出題意図候補を抽出した結果を図 2 に示す。図 2 を見

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int i, j;
6      scanf ("%d", &i);
7      if (i > 0) {
8          for (j = 0; j < i; j++) {
9              printf ("Loop Cnt: " + j + " ¥n");
10         }
11     } else {
12         printf ("Enter value of '1' or more!");
13     }
14     return 0;
15 }
```

図 1 プログラム例

```
1  - Question Intention –
2  Functions: [["main()", "3-15", "int"]]
3  Varoables: [["i", "5-5", "int"], ["j", "5-5", "int"]]
4  Loops: [["for", "8-10"]]
5  Branches: [["if", "7-13"]]
6  Calls: [["scanf", "6-6"], ["printf", "9-9"], ["printf",
7  "12-12"]]
```

図 2 出題意図抽出結果

ると、プログラム中に出現している関数、変数、ループ文、条件分岐文、関数呼出し文が抽出されていることがわかる。これらの要素からは以下の出題意図候補が得られる。

- main()関数を作成すること
- 変数 i を使用すること
- 変数 j を使用すること
- 繰返しには for 文を使用すること
- 条件分岐には if 文を使用すること
- scanf()関数を使用すること
- printf()関数を使用すること

これらの出題意図候補から、教授者が適切な出題意図を選択することで、学習者の解答プログラムを適切に正誤判定することができる。

学習者の解答プログラムは以下のように判定される。

1. テストケースにより出力が正しいか判定する
2. 解答プログラムから出題意図候補を抽出する
3. 解答プログラムの出題意図候補に教授者が設定した出題意図がすべて含まれているかを判定する

3. 実装と評価実験

提案手法を、オープンソースのオンラインジャッジである DMOJ[4] に実装し、有効性を評価する。

評価実験として、出題意図に沿っており出力も正しいプログラム(A)、出題意図に沿っているが出力が正しくないプログラム(B)、出題意図に沿っていないが出力値は正しいプログラム(C)、出題意図に沿っておらずかつ出力も正しくないプログラム(D)の 4 種類のプログラムを用意した。これらのプログラムを従来システムと提案手法を搭載したシステムで正誤判定する。正誤判定の結果を表 1 に示す。従来システムでは、出力のみを確認して正誤判定をするため、プログラム A とプログラム C が正解と判定されている。それに対して提案システムでは、出題意図に沿っているかどうかを正誤判定に加えているため、プログラム C は不正解と判定できている。このとき判定結果は図 3 のように表示される。以上より、提案手法により学生の解答プログラムをより適切に評価することが可能となり、提案手法の有効性を確認することができた。

表 1 評価実験結果

	従来システム	提案システム
プログラム A	○	○
プログラム B	×	×
プログラム C	○	×
プログラム D	×	×

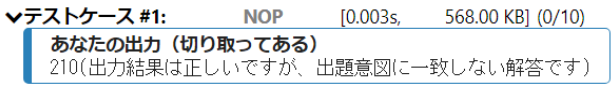

図 3 出題意図に沿っていない場合の判定結果

4. おわりに

本稿では、プログラミング学習支援システムにおいて、学習者の解答プログラムの正誤判定に出題意図を加える手法を提案した。教授者の用意した正解プログラムを分析することで出題意図を抽出し、学習者の解答プログラムにその出題意図が含まれているか否かをプログラムの正誤判定に追加した。評価実験により、提案システムが学習者の解答プログラムをより適切に評価することができることを確認した。

今後の課題として出題意図の拡張が挙げられる。提案手法では、教授者の正解プログラムを基に出題意図を抽出しているので、「条件分岐には if 文を使用すること」という出題意図は抽出できるが、「使用しないこと」という出題意図を設定できない。したがって、「strlen()関数を使用せずに文字列の長さを表示する」のような出題意図を設定できるような拡張が必要となる。

参考文献

(1) 文部科学省：学習指導領域「生きる力」, <URL: https://www.mext.go.jp/a_menu/shotou/new-cs/index.htm> (2023 年 2 月閲覧).
(2) 関澤 俊弦, 河野 一輝: プログラミング教育におけるモデル検査を用いた C 言語プログラムの学修目標の達成度評価に向けて, 情報処理学会論文誌, Vol.63, No.11, pp.1679-1683 (2022).
(3) LLVM Developer Group. : The LLVM Compiler Infrastructure, <URL: https://llvm.org/>.
(4) Guanzhong Chen et al DMOJ:Modern Online Judge <URL: https://dmoj ca/>.