

抽象構文木を用いたプログラミング理解度採点の試み

漆原 宏丞¹ 本多 佑希² 岸本 有生³ 兼宗 進¹

¹大阪電気通信大学 ²四天王寺大学 ³大阪電気通信大学高等学校

1. 緒言

小学校から高等学校まで広くプログラミングが必修化され、教員は生徒の理解度を把握することが求められている。生徒一人ひとりがどこまで理解していて、どこで躓いているか、といった情報は重要である。しかし、教員の目視によるソースコードの確認は、数十人分の確認をミス無く行う必要があり、負担が大きい。

本研究では、生徒の理解度を評価する手法として、生徒のプログラムの構文木からプログラムの内部で使われている構造を検出し、課題で意図した考え方でプログラムが記述されているかを確認するシステムを提案する。

2. 既存の理解度評価手法

プログラムの正誤判定には、出力値を確認する手法と、ソースコードを確認する手法が存在する。前者は情報オリンピック[1]やAtCoder[2]などのプログラミングコンテストでよく使われている。この手法では、プログラム内の処理が正しいかを確認できるが、理解度を段階的に評価することには向いていない。

対して、後者は理解度の段階的な評価を測る手法が提案されている。De-Gapper[3]は、プログラミングの理解に必要な学習ステップの無いように、新しい課題が入っているかを調べるためのツールである。学習のステップを定義し、生徒が作ったプログラムがどのステップに相当するかを確認することで、理解度を段階的に評価することができる。

構文木を使った例としては、抽象構文木レベルで正答プログラムと比較したり、正誤判定を行うシステムが提案されている[4][5]。

3. 設計

3.1 概要

前章で述べたように、プログラムの理解度を評価するために様々な手法やシステムが提案されている。私はその中でも抽象構文木を確認する手法に着目した。プログラムから生成された抽象構文木の部分木を確認することで、そのプログラム内で使われている制御構造を検出

することができる。そのため、問題で意図した構造がプログラム内で使われているかを確認することで、制御構造の理解度を測ることができるのではないかと考えた。システムの概要図を図1に示す。

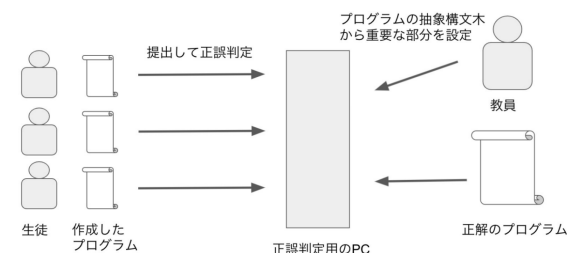


図1 システム概要図

3.2 必要な機能

今回提案する手法をシステムとして動作させる場合、以下のような流れになる。

1. プログラムから抽象構文木を生成する
2. 問題が意図する抽象構文木を生成する(場合によっては、教員が正解とする木を指定する)
3. プログラムから生成された構文木の中に、正解となる構文木が部分木として含まれているかを確認する

そのため、これら3つの機能を検討・実装することにした。

4. 実験と評価

4.1 実験に使用した課題

システムの有効性を確認するため、工学部の大学2年次の開講されるプログラミングの授業課題と、その受講生の提出プログラムを使用する。この授業はPythonで行い、実行環境としてオンライン学習環境のBit Arrowを使用していた。

4.2 生徒のプログラムを評価した結果

問題が意図するであろう構文木を筆者が作成し、各課題のプログラム(計1343個)に適用した。その結果として、構文木にマッチするが出力結果が正しくないものを表1に、その一例を図2に示す。図2のプログラムは、3行目のxをnにし忘れたり、4行目でen(x)ではなくen(3)と

している間違いがある。しかし、制御構造自体は間違っていないので、理解度は足りていると判定すべきだと考える。

また、構文木にマッチしないが出力結果が正しいものは30件あった。この例を図3に示す。図3の課題では、for文を使って解く方法が期待されるが、図3のように組み込み関数を使った回答が散見された。このプログラムは出力結果こそ正しいものの、この課題で確認したかった理解状況が見られないので、理解しているとはいえない。

本システムでは、こういったプログラムを正しく判定することは難しい。そこで、2章で述べた出力値を確認する手法(ホワイトボックステスト、以下WBT)を組み合わせることが考えられる。本システムでOKと判定され、WBTでNGと判定されたものだけを人間が目視で確認することで、少ない労力で生徒の理解状況を調べることが可能になると考える。

表1 構文木にマッチするが出力結果が正しくないプログラムの内容と件数

引数の設定ミス	5
正しくない計算式	27
条件式の記述ミス	10

(母数は提出された回答プログラム全てである1343個)

```
x=int(input('hankei: '))
def en(n):
    return x*x*3.14
print(en(3))
```

図2 表1の引数の設定ミスの一例(半径nの円の面積を返す関数en()を定義する課題。ただしnはキーボードから入力した値を用いる)

5. 結言

生徒のソースコードから構文木を生成し、その部分枝を確認することで、制御構造の理解度を測る手法を提案した。本手法を用いることで、変数名の間違いや単純な計算ミスなど、制御構造とは関係ない部分を無視することができる。

実際に検証したところ、提出プログラム1343個の中で、本システムでOKと判定されたプログラムのうち出力結果が正しくないものは42個

(約3%)となった。今後は、本システムとWBTを組み合わせた判定などを検証する予定である。

```
a=[10, 8, 33, 12, 25]
ave = sum(a)/len(a)
print(ave)
print(max(a))
```

図3(a) 構文木にマッチしないが出力結果は正しい例(配列aの平均値と最大値を求める課題)

```
.*(Loop).*(If).*(Assign)
.*(Loop).*(Assign).*(Add)
.*(Div)
```

図3(b) 図3(a)の課題で作成した構文木(この3つがマッチすればOK判定となる)

・参考文献

- [1] 情報オリンピック.
<https://www.ioi-jp.org/>
- [2] AtCoder. <https://atcoder.jp/>
- [3] 長慎也, 保福やよい, 西田知博, 兼宗進. De-gapper - プログラミング初心者の段階的な理解を支援するツール. 情報処理学会論文誌, Vol.55, No.1, pp.1-12, 2014.
- [4] 小山昂紘, 原田史子, 島川博光. 階層構造化による C 言語 ソースコードの自動採点. In IEICE Conferences Archives. The Institute of Electronics, Information and Communication Engineers, 2012.
- [5] 小川弘迪, 小林亜樹. プログラミング課題の自動採点に向けた構文木上のカーネル法による類似度関数の提案. 第80回全国大会講演論文集, 2018(1), pp.661-662, 2018.