木の編集距離による文単位での C プログラムの定量的評価に向けて

Toward Quantified Evaluation of C Programs at Statement-level Utilizing Tree Edit Distance

達川 航充[*1], 関澤 俊弦[*2]
Wataru TATSUKAWA[*1], Toshifusa SEKIZAWA[*2]
[*1] 日本大学大学院工学研究科
*1 Graduate School of Engineering, Nihon University
[*2] 日本大学工学部
*2 College of Engineering, Nihon University
Email: cewa25007@g.nihon-u.ac.jp

あらまし: 大学における学生のプログラム評価手法については様々な研究が報告されている。代表的な手法の一つは、抽象構文木 (AST) を用いてプログラムの類似性を評価する。本研究では、C 言語の演習課題 1 問の学生 50 名の提出プログラムについて、正規化 AST に対する文単位に基づく木の編集距離による評価を行う。編集距離は理解度の評価指標の一つとなる可能性を示す。
キーワード: プログラミング教育、正規化 AST、文単位評価

1. はじめに

大学のプログラミング教育では、学生は講義内容に沿って演習としてプログラムを記述する。プログラムの記述においては、式と文や、制御構造、関数など様々な言語要素の知識が必要となる。

プログラミング教育におけるプログラム評価については数多くの手法が研究されてきた[1], [2]。他の研究 [3], [4] では、木の編集距離に基づく 抽象構文木 (AST: Abstract Syntax Tree) による評価を報告している。この報告では AST 全体を比較対象としている。また、AST の部分木を扱うことでプログラムの文レベルでの解析を可能にする研究も多数存在する。

本研究は AST に基づく文単位の評価によるプログラミング理解度評価を目的とする。AST を用いたプログラム比較による定量的な評価のため、プログラミング言語の要素とプログラムの構造を扱う。

2. AST の等価性と木の編集距離

AST の等価性とは、2 つの AST が同一か否かを判定することである。AST の等価性にはいくつかの分類がある。比較前に変数などを正規化した後に判定する正規化構造的等価性や限定された入力に対して同一の振舞いを確認することで判定する行動的等価性などがある。

木の編集距離とはある木構造を別の木構造に編集変換するために必要な最小操作数である。ZhangShasha (ZSS) アルゴリズム[5]は、木の編集距離を求めるアルゴリズムであり、編集には挿入、削除、置換の操作が用いられる。ラベル付き順序木 T_1, T_2 に対する木の編集距離 $\delta(T_1, T_2)$ は、次のとおり定義される。

$$\delta(T_1, T_2) = \min\{cost(S) \mid S \text{ is a sequence of edits transforming } T_1 \text{ to } T_2\},$$

ここで、コスト関数 $cost(S)$ は、操作列 S に含まれるすべての編集操作における累積総コストを算出する。

3. 手法

本研究は学生の C 言語についての理解を定量的に評価するため、正規化構造的等価性と行動的等価性を組み合わせた手法[6]を用いる。

ここで、ex_i を i 番目のプログラミング課題とし、解答モデルの C プログラムと提出された C プログラムを、それぞれ p_{ans} と p_{sub} とする。また、p_{ans} と p_{sub} から変換された正規化 AST をそれぞれ AST_{ans} と AST_{sub} とする。これらの正規化 AST、AST_{ans} と AST_{sub}、は文単位で比較される。具体的には、p_{ans} における評価の基準となる文を S とし、対応する部分木を $AST_{ans}[S]$ と表記する。また、$AST_{sub}[T_k]$ を文 T_k に対応する AST_{sub} の部分木と定義する。本手法では、$AST_{ans}[S]$ を選択し、各文 $T_k (k \geq 1)$ に対して AST_{sub} から $AST_{sub}[T_k]$ を抽出する。次に木の編集距離 $\delta(AST_{ans}[S], AST_{sub}[T_k])$ を求める。複数の木の編集距離のうち次の式で定義されている最小の距離 I を理解度の指標として採用する。

$$I = \min\{\delta(AST_{ans}[S], AST_{sub}[T_k]) \mid \text{for each } T_k\}$$

図 1 に、手法の過程を示す。

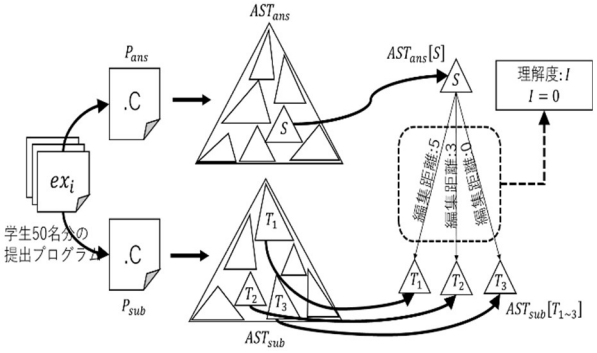

図 1 正規化 AST に基づく手法の過程

本研究では、変数名および、関数名、ブロック文、糖衣構文のうちインクリメント/デクリメント演算子を除く複合代入演算子のみを正規化の対象とする。ブロック文を含む `if`、`while`、`for` などの文は単一文として正規化されるが、ネスト構造は保持される。また、数値と文字列は正規化の対象としない。

4. 評価

本研究を評価するために、日本大学工学部情報工学科の講義「プログラミングの基礎及び演習」で2024年度に課された課題1問に対して、提案手法を適用する。演習問題は、C言語の反復処理の復習問題である。この演習問題に対して、模範解答のプログラムをp_{ans}として、学生の提出プログラムp_{sub}との比較を行う。本研究で用いたプログラミングの基礎及び演習の演習問題文を次に引用する。

「n人の学生(n<1)の試験の点数(整数値 0～100)を読み込み、平均点及び最高点と最低点の人の学籍番号と点数を示すプログラムを作成せよ」

本研究におけるp_{sub}は上記の問題を解いた学生50名のプログラムである。本研究で用いる提出プログラムはすべてコンパイルが通ることを確認している。

対象とした演習課題は復習問題であることから、内容の理解度や定着度合いに着目する。この演習課題の模範解答p_{ans}では `for` 文を用いていることから、本研究では `for` 文同士の比較結果に限定して木の編集距離を評価する。評価のためには木の編集距離の各操作に対してコストを定める必要がある。本研究では、挿入と削除はコスト 1.0 とする。置換のコストとして、変数名の差異による置換は0.1、定数の差異は0.5として、他は1.0とする。なお、同一の置換のコストは0.0である。

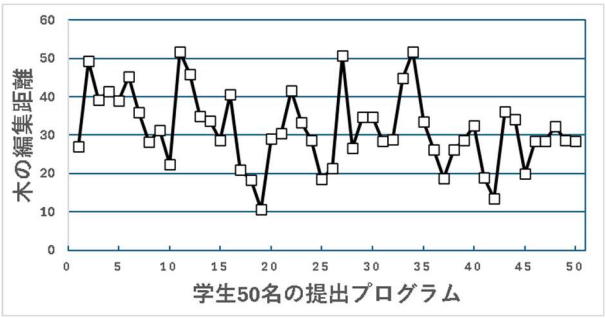

図 2 提出プログラムの木の編集距離

図2に各学生の提出プログラムの木の編集距離を示す。各学生の木の編集距離は、コストに比べると大きい。得られた木の編集距離は最小値が `10.7`、最大値が `51.8` である。この結果は、学生の提出プログラムが多様であるためと考えられる。よって、文単位での正規化 AST の比較による木の編集距離は、学生のプログラムの理解度として有効であると考えられる。

5. 考察

本研究で対象とした演習課題の評価で木の編集距離が大きくなった理由について考察する。本研究の模範解答p_{ans}では、`for` 文内で入出力処理および条件分岐による処理が行われている。学生の提出プログラムでは、条件分岐として `if-else` 文や異なる条件式など様々な解法が見られる。これらの差異により木の編集距離が大きくなったと考えられる。今回の評価で得られた学生全体の木の編集距離の結果はそれぞれ異なっている。この結果より、文単位の木の編集距離は学生のプログラミング理解度を測る指標の一つになる可能性があると考えられる。

6. おわりに

本研究では、演習課題1問の提出プログラムに対して、繰り返し処理に着目して文単位で木の編集距離の評価を行った。個々の学生についての評価では大きな木の編集距離が得られた。学生50名の木の編集距離はそれぞれ異なっている。この結果は、文単位の木の編集距離は学生の理解度を測る指標となる可能性を示している。

木の編集距離は操作の累積コストであるため、個々の操作を特定することはできない。この問題に対して、今後の課題の一つは、編集操作のコストの定義の設定により差異が生じた文を特定できるようにすることである。

謝辞

本研究はJSPS科研費 JP24K15233の助成を受けたものです。

参考文献

(1) J. C. Paiva, J. P. Leal, and A. Figueira, “Automated assessment in ´ computer science education: A state-of-the-art review,” ACM Transactions on Computing Education (TOCE), vol. 22, no. 3, pp. 1–40, 2022.
(2) M. Messer, N. C. Brown, M. Kolling, and M. Shi, “Automated grading ¨ and feedback tools for programming education: A systematic review,” ACM Transactions on Computing Education, vol. 24, no. 1, pp. 1–43, 2024.
(3) Y. Song, C. Lothritz, X. Tang, T. Bissyande, and J. Klein, “Revisiting code ´ similarity evaluation with abstract syntax tree edit distance,” in Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2024, pp. 38–46.
(4) A.-T. P. Nguyen, V.-D. Hoang et al., “Development of code evaluation system based on abstract syntax tree,” Journal of Technical Education Science, vol. 19, no. Special Issue 01, pp. 15–24, 2024.
(5) K. Zhang and D. Shasha, “Simple fast algorithms for the editing distance between trees and related problems,” SIAM journal on computing, vol. 18, no. 6, pp. 1245–1262, 1989.
(6) W. Tatsukawa, and T. Sekizawa, “Statement-level Evaluation of C Programs with Tree Edit Distance on Normalized AST,” 2026 the 14th International Conference on Information and Education Technology, (accepted, to appear).