N-033

AST 解析・変数重み付け・乱数制御を利用した Java プログラミング課題採点システムの構築
Development of a Java Programming Assignment Grading System Using AST Analysis, Variable Weighting, and Randomness Control

岡崎 壮汰† 土屋 誠司‡ 渡部 広一‡
Sota Okazaki Seiji Tsuchiya Hirokazu Watabe

1. はじめに

プログラミングの講義において，受講者から提出されたソースコードの採点は手作業で行われることが多い．採点者は受講者が提出したソースコードを個別に確認し，出力に誤りがないか，ソースコードが指定された条件を満たしているかなど，さまざまな観点から評価を行う必要があるため，大きな負担となっている．

このような背景から，本研究ではプログラミング課題の採点の効率化を目指し，答案を自動で採点するシステムを構築することを目的とする．採点システムは，学生が提出した答案のソースコードから抽象構文木（AST: Abstract Syntax Tree）を生成し，木構造を解析して採点を行う．また，抽象構文木の解析だけでは判断できない出力の確認のために，乱数の値を固定したブラックボックステストや，変数重み付けによる採点時に重要な変数の特定などの機能を追加する．

以上のように，本研究ではプログラミング課題の採点作業を効率化し，教育の質の向上を図るためのシステム開発に取り組む．

2. 既存システム

既存研究である「抽象構文木を利用したプログラミング理解度採点の試み」[1]で提案されている採点システムについて述べる．このシステムは，Python のプログラミング課題の答案のソースコードを採点するシステムであり，システムを構築する言語も Python である．

2.1 既存システムの採点手法

既存システムは，Python の ast モジュールを用いて答案のソースコードから抽象構文木の文字列を生成し，正規表現により抽象構文木の文字列を検索して採点を行う．

既存システムを用いて，例題 A「キーボードで入力された整数 n を引数とし，n の二乗を返す関数を定義せよ．」を採点することを考える．例題 A を採点する場合の採点基準と，採点基準を表す正規表現の例を表 1 に示す．表 1 の採点基準(ア)では input というノードを検索しており，採点基準(イ)では int というノードを検索している．採点基準(ウ)では，FunctionDef というノードの下に Mult というノードがあるかを検索している．FuctionDef というノードは関数定義の記述を表すノードであり，Mult は掛け算を表すノードである．

† 同志社大学大学院理工学研究科
‡ 同志社大学理工学部インテリジェント情報工学科

表 1　例題 A の採点基準を表す正規表現

採点基準	採点基準を表す正規表現
(ア)入力した値を引数として計算する	.*(input).*
(イ)入力した値を数値に変換する	.*(int).*
(ウ)関数定義の中で 2 乗の計算を行う	.*(FunctionDef).*(Mult).*

2.2 既存システムの問題点

抽象構文木の文字列を正規表現によって検索する場合，ノードの名前が採点基準で指定した文字列と完全一致しているかを判断できない．例えば，ノード int を正規表現「.*(int).*」で検索する場合，ノード print も該当するため，誤った採点結果が出力される可能性がある．

また，ソースコードから作成した抽象構文木の文字列を正規表現によって検索する場合，抽象構文木の階層構造を正しく判断できない．例として表 1 の採点基準(ウ)の正規表現は，抽象構文木の文字列の前後関係しか判断できないため，関数定義部分の記述が終わったあとに掛け算が含まれていても，一致したと判断されてしまう．

また，既存システムで採点対象となっている問題は答案のソースコードの行数が少ない問題だけであるため，既存システムは複雑な問題に対応できない可能性がある．

3. 提案手法

本研究では，既存システムの採点手法や問題点を参考にし，Java のプログラミング課題を自動で採点するシステムを構築する．採点対象となる答案の言語だけでなく，採点システムの言語も Java に変更する．Java は同志社大学理工学部インテリジェント情報工学科のプログラミングの授業で用いられているほか，容易に使用できる構文木操作ライブラリ（Eclipse JDT）が存在しているため，本研究では Java の課題を自動採点するシステムを構築する．この方針をもとに，既存システムから変更する点を以下に示す．

3.1 抽象構文木の文字列を木構造として認識

既存システムでは答案のソースコードから作成した抽象構文木を文字列として認識して採点していたが，提案手法のシステムでは抽象構文木を木構造として認識して採点する．そのため，既存システムでは正規表現で表していた採点基準を，提案手法のシステムでは木構造で表す．具体的な採点方法は，答案のソースコードから生成した抽象構文木の木構造の中に，採点基準となるノードまたは木構造が含まれているかを検索するという方法である．例として，

例題 A の正解の答案のソースコードから作成した抽象構文木の木構造を図示した結果を図 1 に示す.

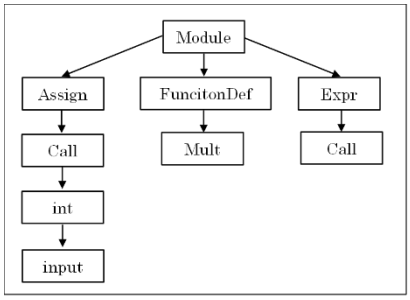

図 1　答案のソースコードから作成した抽象構文木の木構造

表 1 の採点基準(ア)と(イ)の正規表現の検索は，図 1 においてそれぞれノード「input」，「int」を探すことと同じである．採点基準(ウ)の正規表現の検索は，図 2 の木構造が図 1 の部分木として含まれるかを探すことと同じである．

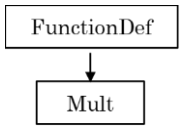

図 2　3 つ目の採点基準を表す木構造

3.2 乱数値を制御したブラックボックステストを導入

採点精度を向上させるとともに，複雑なプログラミング課題も採点できるようにするため，ブラックボックステストを導入する．答案に乱数生成の処理が含まれる場合，乱数の値によってブラックボックステストの出力が変わるため，テストの際には採点システムが乱数の値を制御する．

3.3 変数重み付けによる採点時に重要な変数の特定

抽象構文木の解析とブラックボックステスト以外の採点手法として，採点時に重要となる変数の値を確認する手法を導入する．プログラミング課題において変数名を指定することは実用上困難であるため，変数名に依存せず採点で重要な変数を探す必要がある．そのため，変数が重要であるかを判断する指標（再代入されていない，コンソール出力に使用されるなど）をもとに，重要度を重み付けすることにより採点で重要な変数を特定する機能を追加する．

3.4 プログラミング課題を複雑なものに変更

既存システムで採点対象となっている問題は，答案のソースコードの行数が少ない問題だけである．そのため，提案手法のシステムは答案の行数が多い複雑な問題も採点できるようなシステムに改良する．

4. 評価と結果

実験協力者(2 名)が作成した問題(6 問)を別の実験協力者(9 名)に解いてもらい，得られた答案(38 個)に対するシステムの採点結果と実験協力者(3 名)の採点結果の平均値を比較することによりシステムを評価した．採点は 10 点満点で行った．

4.1 評価指標

評価指標として，一致率，不一致率，ウィルコクソンの符号付き順位検定[2]を用いる．

一致率とは，全体の答案数に占める，システムの点数と人間の点数の平均の差が 1 以下の答案数の割合である．不一致率とは，全体の答案数に占める，システムの点数と人間の点数の平均の差が 1 を超える答案数の割合である．

また，ウィルコクソンの符号付き順位検定とは，2 組の標本の対になったデータの差が 0 に等しいかを検定するノンパラメトリック検定である．本システムの評価時は，帰無仮説を「システムと人間の採点結果の代表値の差が 0 である」とし，対立仮説を「システムと人間の採点結果の代表値の差が 0 でない」とする．ウィルコクソンの符号付き順位検定は 2 種類の採点結果の傾向が一致しているかを確認する検定であり，採点結果の値の一致性を確認することはできないため，本システムでは参考情報としてこの検定の結果を確認する．

4.2 評価結果

6 問すべての答案について，一致率は 65.8%，不一致率は 34.2%であった．また，ウィルコクソンの符号付き順位検定の結果，p 値は 0.697 であった．よって，有意水準 5%で帰無仮説「システムと人間の採点結果の代表値の差が 0 である」を採択する．

5. 考察

ウィルコクソンの符号付き順位検定の結果，帰無仮説が採択されたため，システムの採点結果と人間の採点結果には有意差がないと判定された．ウィルコクソンの符号付き順位検定の性質上，厳密な値の一致は確認できないが，採点結果の傾向には有意差がないため，システムの性能は良いと考えられる．

個々の答案の採点結果の精度に着目すると，人間の採点結果とシステムの採点結果が不一致である答案の割合が高い．本システムで採点を行う際，人間用の採点基準をもとにシステム用の採点基準を決定したが，システム用の採点基準の決定手法に改善が必要であると考えられる．

6. まとめ

本研究では，Java のプログラミング課題の答案を自動で採点するシステムを構築することを目指した．システムは構築できたが，様々な課題や答案が存在するため採点システムの構築は難しく，採点基準の内容を正確に判断できていない部分もあった．システムだけで採点を完結させることは非常に困難であるため，まず人間の採点作業を補助するシステムを構築することが，システムの精度向上や採点の効率化に繋がると考えられる．

参考文献

[1] 漆原宏丞，本多佑希，岸本有生，兼宗進，“抽象構文木を利用したプログラミング理解度採点の試み”，情報処理学会「研究報告コンピュータと教育」，Vol.2021-CE-162，No.16，pp.1-6，2021

[2] 村上秀俊，“統計解析スタンダード ノンパラメトリック法”，株式会社朝倉書店，2015，p.32