学習者プログラム半自動評価システムへの動的解析の導入

Dynamic Analysis for a Semi-automatic Program Evaluation System

王　子頁[*1]，小暮　悟[*1]，小西　達裕[*1]，伊東　幸宏[*2]
Ziye WANG[*1], Satoru KOGURE[*1], Tatsuhiro KONISHI[*1], Yukihiro ITOH[*2]
[*1] 静岡大学大学院情報学研究科　[*2] 静岡大学
[*1]Graduate School of Informatics, Shizuoka University　[*2]Shizuoka University
E-mail:gs11302@s.inf.shizuoka.ac.jp

あらまし：我々は先行研究において、プログラミング教育における教師支援を目的としてプログラムの静的解析によるプログラミング半自動評価器を持つシステムを構築してきた。本稿ではプログラム半自動評価器に動的解析を導入することで評価精度を向上させる手法を提案する。はじめに先行研究における問題点の調査結果を報告する。続いて問題点を解決するための動的解析手法を述べる。最後に動的解析結果の表示方法を示す。
キーワード：教育支援システム，プログラム評価，初等プログラミング教育，動的解析

1. はじめに

先行研究の学生プログラム半自動評価器[1]では学生の提出したプログラムを静的解析し、教師が指定した標準アルゴリズムとの一致度に基づいて選別することで教師の負担を軽減することができた。このシステムの評価実験を行い、情報系学部 2 年生向けプログラミング演習科目で提出されたプログラムを評価させた。結果のうちシステムが一定以上の誤りを含むと判定したものが 233 例あった。そのうち教師が正しいと判定したプログラムを、正解アルゴリズムの実装方法と異なるために間違いと判定した例 (以下別解と呼ぶ)が 36 例あった。このような例はプログラムの表層から評価する静的解析では評価が難しいが、プログラムの実行プロセスを解析する動的解析では正しい評価が可能な場合がある。そこで本研究では、現行システムに動的解析を導入することで評価精度を向上させることを目的とする。

本研究で構築するシステムは静的解析と動的解析を併用し、次の特徴を持つ。

①先行研究における静的解析から取得した変数の対応関係を利用し、動的解析を行う。これにより変数の対応関係の仮説を多数生成する必要がなく、計算量の爆発を回避できる。

②演習問題中で使用する変数名・関数名を指定しなくとも評価できる。これにより自由度の高い演習を実現する。

③動的解析は実行結果 (プログラムの出力)だけでなく、実行途中の変数値や、ループ実行回数と分岐状況などの実行プロセスもチェックできる。

④動的解析の結果として様々なテストケースにおいて均一な挙動か否かを提示できる。

⑤動的解析は標準アルゴリズムとの完全一致のみを検出するのではなく、各テストケースにおいて一定の差分を有する場合も検出できる。

⑥テストケースはシステムにより半自動的に生成することができ、教師の負担を更に軽減できる。

プログラムの動的解析をプログラミング教育に利用する手法にはいくつか先行研究がある[2][3]。高野ら[2]は、学生プログラムを教師が予め設定したルールに従って静的解析し、更に学生プログラムの出力結果と正解を比較することで学生プログラムを評価する手法を提案している。しかし、この動的解析は静的解析の結果を利用していない。また、プログラムの出力結果だけを評価する。中島らの手法[3]では、教師が用意した正解プログラムと学習者の答案プログラムに複数のテストケースを与え、その実行プロセスを比較する。しかし、この手法では演習問題中で使用する変数名・関数名を教師が予め指定しなければならず課題の自由度が低くなっている。また、[2][3]の手法はいずれもテストケースを教師が作成しなければならない。

2. 現行システムにおける問題点

前述の別解 36 例のプログラムを分析し、現行システムの静的解析手法で正しく評価できない 3 つの典型的パタンを発見した。

(1) 同じ意味のループだが実装方法が異なる

本質的に同一の反復処理であっても、制御変数の定義、反復処理で書き換えられる変数の定義、ループの脱出条件などプログラムの表層としては多様な記述が可能である。そのため、静的解析では同じ論理の反復処理を同一と判定できないことがある。これは動的解析によりループの反復回数を比較したり、対応する変数の値に常に一定の差分があることを検出することで正しい判定ができる可能性がある。例えば配列中の数値の総和を求めるプログラムでは実装方法の差異があっても反復回数は配列の要素数に一致すると考えられる。

(2) 条件式の真偽及び判定後の処理が逆

本質的に同一の条件分岐だが、条件式の真偽及び判定後の処理が互いに逆に定義されている二つ

のプログラムを同一であると判定できない。

(3) 関数化のくくり方が異なる

同一の処理手順を持つプログラムだが、関数化された部分に差異があるプログラムを同一であると判定できない。

3. プログラム評価への動的解析の導入

本研究の動的解析の観測範囲は静的解析で対応が付かなかった領域同士とする。また、観測対象の変数は静的解析で対応が付いている変数同士とする。ただし、他の観測したい変数があれば教師はシステムに指定できる。もう一つの観測対象は実行プロセスであり、ループの実行回数と選択構造における条件式の真理値を観測する。

観測は以下の二種類の観測ポイントで行う。

(a) 変数の挙動を観測するための観測ポイント

観測範囲を脱出する時点及び出力文の実行時点に設定する。

(b) 実行プロセス観測のための観測ポイント

for（while）文と if（else）文のブロックの先頭に設定する。比較対象のプログラムをそれぞれ実行し、観測ポイントの実行シーケンスを比較する。

テストケースの自動生成については大塚らの手法 [4]を参考に検討中である。具体的にはブラックボックステスト技法に基づいてテストケースの生成を行い、ホワイトボックステスト技法のブランチカバレッジ技法によりテストケースの網羅性を検証することを考えている。

4. 動的解析結果の表示方法

4.1 変数の比較結果の表示

ある観測ポイントにおける変数値の比較結果の提示方法を図 1 に示す。

標準アルゴリズム	学生プログラム	対応状況	差分状況
X	X1	[青] 詳細	==
Y	Y1	[黄] 詳細	*d d=1/2
Z	Z1	[赤] 詳細	

図 1　変数対応状況表示図

全てのテストケースに対する比較結果の均一性を色で表示する。青は全てのテストケースにおいて比較結果が完全に一定という意味である。閾値以上の比率で一定の場合は黄色で表示する。一定でない場合は赤で表示する。「一定」には比較結果において「完全一致」、「決まった差分がある」の二つの状況があり、それぞれ「==」、「+/-/*/÷ d」と表示する。例えば図 1 の変数 Y と Y1 の比較結果の対応状況は黄色になっており、差分は*d、d=1/2 である。よって、ほとんどのテストケースに対して比較結果が一定しており、Y1 が Y の 1/2 倍になっていることを示している。

また、教師がどのようなテストケースの場合に 1/2 倍の差分ではないか調査したい場合、詳細ボタンをクリックすると各テストケースにおける変数値を見ることができる。

4.2 実行プロセスの比較結果の表示

実行プロセスの表示方法としてはループ・if 文のそれぞれに含まれる観測ポイント(図 2 では PT、PS)の、実行回数・条件式の真理値の比較結果を表示する。セルの色の意味は変数の場合と同様に各テストケースに対する結果の均一性を表している。また、ループの実行回数の差分や条件式の真理値が逆転していることを再右列に表示する (図 2)。

標準アルゴリズム	学生プログラム	対応状況	差分状況
ループ			
PTm (ループA)	PSx (ループa)	[青] 詳細	==
PTn (ループB)	PSn (ループb)	[黄] 詳細	+d d=+1
PTz (ループC)	×	[赤] 詳細	
if文			
PTx (if文A)	PSy (if文a)	[黄] 詳細	==
	PSv (if文b)	[黄] 詳細	==
PTy (if文B)	PSm (if文c)	[青] 詳細	逆転
×	PSi (if文d)	[赤] 詳細	

図 2　実行プロセス対応状況表示図

5. むすび

提案したシステムは現在実装中であり、既存のデバッガーGDB の機能を利用して動的解析を実現している。今後は演習で提出された実プログラムを対象に評価実験を行う予定である。

参考文献

(1) 鈴木浩之,小西達裕,伊東幸宏：“抽象的データ構造を含むアルゴリズム表現に基づくプログラミング評価支援システムの構築”,教育システム情報学会誌,Vol.24, No.3, pp.682-692 (2007)

(2) 高野佑一,海谷治彦,海尻賢二:“拡張可能な診断記述項目を用いた多視点からのプログラム診断システム”,情報処理学会, 第 69 回全国大会講演論文集, pp.449-450 (2007)

(3) 中島秀樹, 高橋直久, 細川宣秀：“プログラミング学習のための QA サイクル-受講者の習得度に応じた問題自動提示メカニズム-”,電子情報通信学会論文誌, Vol.J88-D-I No.2, pp.439-450 (2005)

(4) 大塚俊章,萩野富二夫：“ソフトウエアテスト技術(特集 ソフトウェアエンジニアリング)”,Unisys 技報,Vol.27, No.2, pp.164-182 (2007)