模索痕跡の分析に基づいたコンパイルエラーに表出しない躓きの推定の試み

An Attempt to Estimate Impasses Not Appearing in Compilation Errors Based on the Analysis of Exploratory Traces

川崎 満広[*1]，大波 奨[*1]，大沼 亮[*2]，中山 祐貴[*3]，神長 裕明[*1]，宮寺 庸造[*4]，中村 勝一[*1]
Mitsuhiro KAWASAKI[1], Sho ONAMI[1], Ryo ONUMA[*2], Hiroki NAKAYAMA[*3], Hiroaki KAMINAGA[*1], Youzou MIYADERA[*4], Shoichi NAKAMURA[*1]
[*1] 福島大学 共生システム理工学研究科／共生システム理工学類
[*1]Department of Computer Science and Mathematics, Fukushima University
[*2] 津田塾大学 学芸学部 情報科学科
[*2] Department of Computer Science, College of Liberal Arts, Tsuda University
[*3] 山形大学 地域教育文化学部
[*3] Faculty of Education, Art and Science, Yamagata University
[*4] 東京学芸大学 教育学部
[*4]Faculty of Education, Tokyo Gakugei University
Email: {kawasaki,onami}@cs.sss.fukushima-u.ac.jp, nakayama@e.yamagata-u.ac.jp, r.onuma@tsuda.ac.jp, {kami, nakamura}@sss.fukushima-u.ac.jp, miyadera@u-gakugei.ac.jp

あらまし：プログラミング演習における的確な指導のためには，学習者の行き詰り等の状況を上手く把握することが重要であるが，多数の学習者に少数の教授者で対応する制約下では，容易ではない．本研究では，プログラミング演習における学習者による模索（ソースコード中のある部分を書いたり消したりする様子）に注目し，その分析に基づいて行き詰まりを自動推定する手法の開発を目指す．本稿では，行き詰まり把握支援の概要を示し，コーディング過程における模索痕跡の抽出について述べる．
キーワード：模索痕跡，インデント整形，エリア分割，ロジック構成

1. はじめに

近年，大学はもとより，小学校から高校まで多くの教育機関でプログラミング教育が実施され，支援ニーズが高まっている．的確な指導のためには，学習者の行き詰り等の状況を把握することが重要である．しかし，プログラミング演習では一般に，多数の学習者に対して，少数の教員と TA で対応する必要があるため，個々の学習者の行き詰まりの把握には限界がある．

これに対して，プログラミング演習における学習状況の把握を支援するシステム(1)やプログラミング演習者の行き詰り自動検出(2)などが報告されている．これらは，クラス全体の状況と個々の学生の状況の把握を視野に入れた実践的な取り組みである．しかし，コンパイルエラーの分析による状況提示の割合が大きく，コンパイルエラーとして表出し難いロジック構成面の行き詰りの把握には対応していない．また，論理面のエラーを対象として行き詰まり箇所を特定する手法が報告されているが，現状では対応可能な行き詰まりが限定的である．

本研究では，プログラミング演習中の学習者によるコンパイル履歴とソースコード間の距離推移の分析に基づいて，コンパイルエラーとして表出しないロジック構成面の行き詰りを推定する手法を開発する．特に，模索痕跡の分析に基づいて，行き詰まりが生じている要素をあわせて抽出する．これにより，教授者によるプログラミング演習中の行き詰まりの把握に対する新たな支援の可能性を示す．

2. 問題点と方針

2.1 問題点

本研究では，プログラミング演習における問題点のうち，主に以下の 2 つに焦点を当てる．

（問題点 1）少数の教授者が多数の学習者それぞれの行き詰りを把握することには限界がある．

（問題点 2）教授者は，コンパイルエラーとして表出しない行き詰まりと，その要素を把握することが困難である．

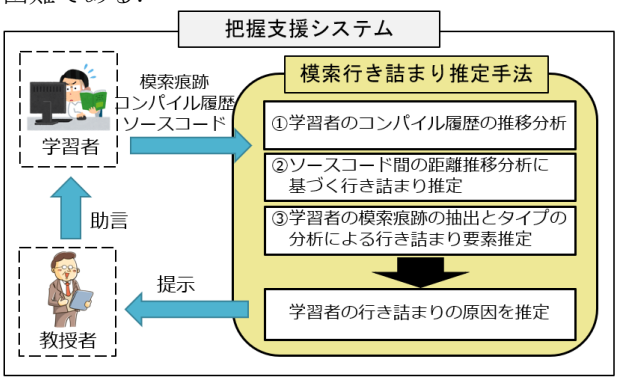

図 1 支援システムの概要

2.2 方針

本研究では，学習者のコンパイル履歴と，ソースコード間の距離推移分析に基づいてロジック構成面の行き詰りを推定する．あわせて，学習者の模索痕跡の分析に基づいて行き詰まりの要素を抽出する手法を開発する（問題点 2 への対応）．

その上で，学習者のコーディング作業を監視し，その分析に基づいて行き詰りが疑われる学習者をその要因と共に教授者に提示するシステムを開発する（問題点 1 への対応）．

3.　コーディング過程における模索痕跡

3.1 模索痕跡

本研究では，学習者がソースコード中の特定の箇所について，「書いたり消したり」を繰り返す行為を模索と呼ぶ．また，この書いたり消したりした箇所とその履歴を模索痕跡と呼ぶ．

3.2 模索のタイプ

研究の第一段階として，以下のタイプに焦点をあてる．

1) ループ骨格：ループの全体像を形作れず，処理部分で模索が行われているタイプ．
2) 条件分岐骨格：条件分岐の全体像を形作れず，処理部分で模索が行われているタイプ．
3) 動作確認
   - 3-1) ループ制御：ループの土台となる，初期化式，継続条件式，変化式などの模索が行われているタイプ．
   - 3-2) 分岐制御：条件分岐の土台となる，分岐条件式で模索しているタイプ．
   - 3-3) 出力確認：変数の動きがイメージできず，標準出力関数などで模索しているタイプ．
4) その他：上記以外の模索．

4.　ロジック構成面の行き詰まりの推定

4.1 行き詰まり要素の推定手順

学習者の行き詰まりを以下の手順で推定する．

1) 学習者のコンパイル履歴の推移分析による 1 次候補の抽出
2) ソースコード間の類似度推移分析に基づくロジック構成面の行き詰まり推定
3) 学習者の模索痕跡の抽出とタイプの分析による行き詰まり要素推定

4.2 ソースコードのエリア分割

エディタ上で書かれたソースコードに変更が加わる度にローカル履歴として保存・蓄積をする．その上で，蓄積されたソースコード間にどの要素で差分が認められるかを判定するために，ソースコードをいくつかのエリアに分割をする．この際，初学者はインデントを効かせてプログラムを書くことに慣れていないため，制御構造等のエリアが不明瞭であることがある．そこで，ソースコードをテンプレート化するためにインデント整形を施す．インデント整形とは，インデントの階層レベルを構造に合わせて自動的に調整し，改行や括弧の位置をルールに則って統一するものである．これにより，エリアの分割の精度を確保する．

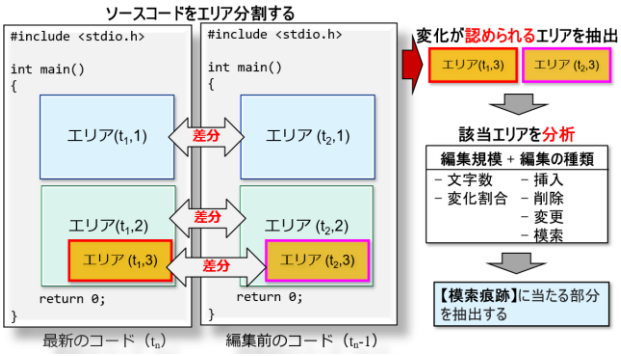

図 2 模索痕跡の抽出

エリアの分割は 2 段階で実施する．まず，中括弧で囲まれている箇所をひとつのエリアとする．次に，予約語とその予約語より高いレベルのインデントを持つ行を同じエリアとする．

4.3 エリアの差分抽出

蓄積された最新のソースコード t_n と編集前のソースコード t_n-1 の差分を抽出し，データベースに蓄積する（図 2）．この際，以下の情報を抽出する．

1) 変更前後の文字（文字列）
2) 1)が変更前後どちらかを示す記号
3) 変更のあった行番号

4.4 模索痕跡の抽出

差分が認められたエリアに対して，タイプミスや単なる書き換えを模索痕跡の要素として抽出することを避けるために，編集規模と編集の種類を分析する．これを基に，模索痕跡が抽出されたエリアを模索箇所として抽出する．

4.5 模索タイプの分析

模索箇所の中でより多く模索が行われている行に注目する．その行が構造を形作る行か構造の処理命令に関する行かを判別することで，その模索のタイプを推定する．この模索タイプとロジック構成面の行き詰まり推定の結果をあわせて分析することで，行き詰まり要素を推定する．

5.　おわりに

本稿では，行き詰まり把握支援の概要とコーディング過程における模索痕跡の抽出について述べた．

今後は，プロトタイプシステムを実際のプログラミング演習に適用した実験を重ね，提案手法の有効性検証と改善を進めたい．

参考文献

(1) 加藤利康，石川孝，“プログラミング演習のための授業支援システムにおける学習状況把握機能の実現”，情報処理学会論文誌, Vol.55, No.8, pp.1918-1930, (2014).
(2) 浦上理，長島和平，並木美太郎，兼宗進，長慎也，“プログラミング学習者のつまずきの自動検出”，情報処理学会研究報告,研究報告コンピュータと教育, Vol. 2020-CE-154, No.4, pp.1-8, (2020).