K-025

問題文の切り分けによる学習者の機能分割能力の評価
Enclosing Assignment Texts to Evaluate Function Decomposition Ability of Learners

板戸 陽子 † 小山 昂紘 ‡ 原田 史子 † 島川 博光 †
Yoko Itado Takahiro Koyama Fumiko Harada Hiromitsu Shimakawa

1. はじめに

多くの大学において，学習者にプログラミングを教えるために，座学と演習の講義を設けている．プログラミングに必要な能力は，文法知識，コンピュータの知識，制御構造に落としこむ力である．文法知識とコンピュータの知識は座学で教えるが，制御構造に落としこむ力は実際に問題を解くことで身に付く能力なので，演習を通して身につけさせる必要がある．しかし，現在の演習では，学習者がどの程度の制御構造に落としこむ力を得たかを評価する機構が乏しい．学習者がどの程度の制御構造に落としこむ力を得たかを評価することで，指導者はその学習者にあわせた指導が可能となる．

本論文では，学習者が出題者の意図通りに機能分割ができるか評価する手法を提案する．これにより，学習者の制御構造に落としこむ力を指導者が把握できる．

2. プログラミングにおける機能分割の把握

2.1 プログラミング教育の問題点

現在のプログラミング演習では，自然言語で書かれた問題文を学習者に与える．指導者は，どのような処理の流れでその問題を解くべきか考えて，出題している．学習者はこの問題文を読み解き，処理の流れを考えた後，ソースコードを作成する．ところが，処理の流れを考えられない学習者は，どこからソースコードを作成すれば良いか，解らなくなる場合がある．その結果，処理の流れを考える前にソースコードを作成しはじめてしまい，計画性のないプログラミングをすることになる．それにより，本来の出題意図に即した学習ができなくなる．

学習者は処理の流れが解らないとき，試行錯誤を繰り返してソースコードを作成する．同じ試行錯誤を経た学習者には，同じ指導が有効と考えられる．しかし，現在の演習では，提出物としてソースコードのみを求める問題が多い．そのため指導者は，学習者がどのような試行錯誤のもとに処理を切り分けたかを把握できない．それにより，学習者を指導するさいに，学習者を適切に分類し，指導をすることが困難である．

2.2 機能分割の重要性

プログラミング演習において，教員は問題文を与えている．この問題文は自然言語で書かれており，学習者が作るべきソースコードの内容が定義されている．しかし，ソースコードは処理の流れに沿って書くべきであるため，学習者は問題文から処理の流れをイメージして，ソースコードを書きはじめる必要がある．処理の流れをイメージするには，まず，どのような処理をするべきか，処理を切り分ける．そして，切り分けた処理を並び替え，適切な流れを作る．このように処理を切り分けて考えることを機能分割という．

以下に，機能分割が必要な演習課題の例を示す．

> うるう年なら返り値 1 を，そうでないなら 0 を返す関数 int leapYear(int year) を作る．この関数を用いて，2001 年から 2999 年までの年で，leapYear 関数の返り値を表示する．また，うるう年が何年あるか計算し，表示する．なお，うるう年とは，4 で割り切れる年のことである．ただし，100 で割り切れる年はうるう年ではなく，400 で割り切れる年はうるう年である．

この演習課題の場合，leapYear 関数はうるう年であるか判定する機能のみを持つべきである．そのため，leapYear 関数の返り値の内容は main 関数で表示すべきである．しかし，関数内で表示をする学習者がいる．それは表示する機能を関数の処理から切り分けていないためと考えられる．このように，機能分割ができていない学生はソースコードを記述しても，指導者の意図とは異なるソースコードを作ることになる．

2.3 既存研究

文献 [1] では，擬似言語を生成してからソースコードを書かせることで，段階的詳細化を支援している．しかし，擬似言語の文法を新たに習得する必要があり，新規学習者への負担が大きい．また，擬似言語からプログラミング言語の自動生成も支援しているため，学習者がプログラミング言語を学ぶことを妨げてしまうおそれがある．

文献 [2] では，生徒が提出したソースコードを解析することで，どのようなアルゴリズムに基いて書かれているかを評価する．しかし，学習者がソースコードを完成させるまでは解析することができないため，機能分割をする段階で行き詰る学習者には対応できない．また，提出したソースコードのみを評価するため，関数化など，機能分割の方法によってソースコードが変化する問題以外では，学習者がどのように機能分割してしまったかを推定することはできない．

3. 問題分割による機能分割能力の評価

3.1 学習者の機能分割能力の評価

本論文では，学習者が出題者の意図通りに機能分割ができるか評価する手法を提案する．本手法では，ソースコードを記述する前に，問題文を切り分けさせる．その処理ごとにソースコードが記述できるか，穴埋め問題を出題する．これらの問題の解答結果を分析することで，学習者の機能分割能力を評価することができる．

本手法の全体図を図 1 に示す．はじめに，指導者は問題文を機能分割する問題を学習者に出す．この問題を切り分け問題とする．切り分け問題では，まず学習者に機能のまとまりごとに，問題文を切り分けさせる．次に，各まとまりを処理と考え，処理を正しいと思う順に入れ替えさせる．その結果を模範解答と比較することで自動採点し，間違っている場合は学習者に通知する．間違いが通知された学習者は，再び切り分け問題を解き，正解

† 立命館大学情報理工学部
‡ 立命館大学大学院理工学研究科

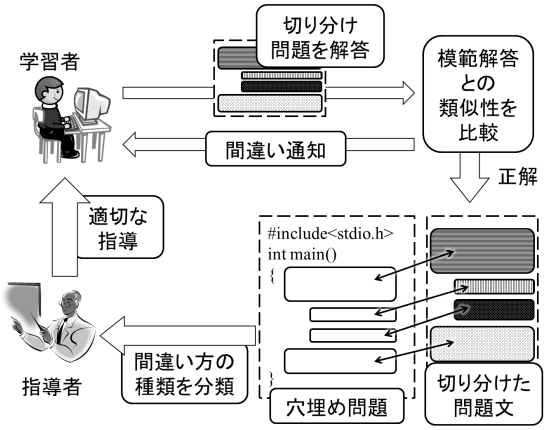

図 1: 手法の全体像

するまでこれを繰り返す．この時に切り分けられた 1 つの処理の区切りをブロックと呼ぶ．

切り分け問題に正答した学習者は，ブロックごとにソースコードを記述する穴埋め問題を解く．この穴埋め問題をブロック別穴埋め問題とする．そして，模範解答と学習者が提出した結果をブロックごとに組み替えて採点することで，ブロックごとに採点する．

ブロックの切り分け問題と各ブロックに対する記述結果と双方の正誤の変遷をもとにして，学習者の特性を把握する．その結果，指導者は学習者がどのような間違いを辿りやすいのかを把握することができる．それにより，学習者ごとに適切な指導が可能になる．

3.2 切り分け問題に対する正誤判定

教員は，切り分け問題に対して，模範解答を用意する．システムは模範解答と生徒が提出した解答を比較し，その解答が一致しているか判定する．そのさい，解答と一致しているかを見るだけでなく，編集距離を用いて間違っている箇所を抽出する．間違いの種類としては，以下の 3 つが考えられる．

- 切り分けるべき箇所が切られていない場合
- 切り分ける必要がない箇所が切られている場合
- 並び替えの順番が入れ替わっている場合

さらに，同じ種類の間違いでも，間違っている箇所に着目することで，間違いの深刻度を測る．

同じ間違いをした学習者でも，間違いをすぐに直すことができた学習者と，試行錯誤をして直した学習者を区別する必要がある．そのため，複数回自動採点した結果をもとにして，学習者ごとに正誤の遷移を分類する．それにより，学習者がどのような切り分けの間違いをし，どのように解決したのかを把握することができる．その結果を蓄積することで，指導者は学習者の切り分けに対する傾向も分類することができる．

3.3 ブロック別穴埋め問題の採点方法

ブロック別穴埋め問題の採点方法を図 2 に示す．学習者は対応するソースコードを過不足なく記述できる必要がある．そのため，ブロック別穴埋め問題では，切り分けたブロックそれぞれに対して，機能が一致しているソースコードを記述できるかを評価する．

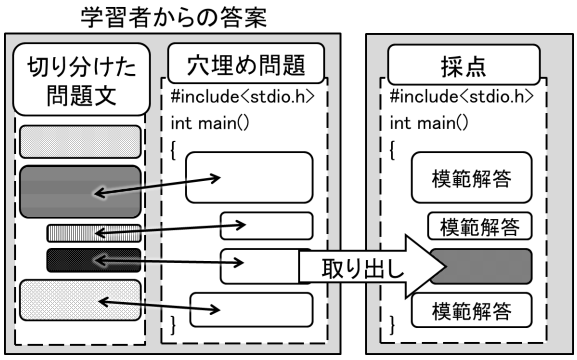

図 2: ブロック別穴埋め問題

はじめに，学習者が提出した穴埋め問題の解答の 1 ブロックを取り出し，その機能と一致する模範解答のブロックと入れかえ，実行する．次に，その実行結果を模範解答の実行結果と比較することで，ブロックごとにそれぞれの処理が正しく書かれているかを見ることができる．

例えば，2.2 節に上げた例のように，leapYear 関数内でうるう年であるか表示をしている学習者がいる．この場合，関数部分を取り出し，模範解答に埋め込み，実行すると，同一のメッセージが 2 回表示される．また，main 関数内の本来表示をするべき場所を取り出して採点した場合には，何も表示されない．これにより学習者が，何をどこで表示するべきかを問題文から読み解けていないことが判る．

この手法では，関数化しない部分でも処理が異なれば，別のブロックと考え，各ブロックごとにコーディングさせる．そのため，関数をもたない問題においても，処理を切り分けて，過不足なく機能を実現できるかを判定できる．これにより，学習者の機能分割能力を演習の初期段階で評価できる．

間違いを犯した学習者が，ブロック別穴埋め問題を繰り返し修正する過程を調べる．その結果，同じ間違いをした学習者でも，間違いをすぐに直した学習者と，試行錯誤をして直した学習者をパターン分けできる．それにより，学習者の試行錯誤のパターンに合わせた指導が可能となる．

4. おわりに

本論文では，ソースコードを記述する前に，問題文を切り分け，処理ごとの穴埋め問題を出題することで，学習者が出題者の意図通りに機能分割ができるか評価する手法を提案した．今後は，本手法の有用性を検証するために，本手法を用いた演習と評価をする予定である．

参考文献

[1] 新開 純子，炭谷 真也，“プロセスを重視したプログラミング教育支援システムの開発 ”日本教育工学会論文誌 31(Suppl.), pp45-48, 2008-02-10

[2] 小西 達裕，鈴木 浩之，伊東 幸宏，“プログラミング教育における教師支援のためのプログラム評価機構”電子情報通信学会論文誌. J83-D-I(6), pp682-692, 2000-06-25