プログラミング学習における対話式誤り修正支援システムの提案
―学習者の理解度と誤りの分類に着目して―

2020SE005 早川 大貴　2020SE078 山口 隼世　2020SE080 山本 大貴
指導教員：蜂巣吉成

1　はじめに

プログラミング学習において，学習者は演習問題に取り組む中で正しい実行結果が得られないことがある．コンパイルは成功するが実行結果が適切でない場合は誤り箇所を知ることが難しい．誤り箇所がわからないまま一人で試行錯誤する状況が想定され，学習意欲の低下を招いている．また，学習者自身が誤りについて理解できていない場合は，指導者に質問すべき内容がわからない状態に陥る．学習者は指導者に質問できれば解決できるが，指導者の人数は限られるので容易に質問できない場合がある．

本研究は，潜在的な誤りを対象に，対話システムを通して誤りを認識させるよう誘導し，ソースコード修正を支援する方法を提案をする．潜在的な誤りとは，文法的な誤りではないがソースコードに誤りと疑われる記述である．潜在的な誤りの一部は，コンパイラにオプションを指定することで警告 (warning) として検出できる．

本研究の技術的課題は，学習者に答えに繋がるヒントをすぐに提示しないような段階的なヒントの提示方法である．段階的な質問にすることで冗長な質問が出てくる可能性がある．それらを排除するために学習者の理解度や誤りの種類によって，提示する質問を変えられる仕組みにする必要がある．

これらの解決のために，次の 2 点をアプローチとする．

1. 誤り推定ツールによって検出された誤りに対応する情報を用いた質問にする
2. どのような質問が有効か分析し，理解度に合わせた質問にする

誤り推定ツールとは，誤り検出ツールと構文解析器を含めたツールである．誤り検出ツールとは，誤り箇所や誤り種類を検出・特定するために使用するツールを指す．誤り推定ツールを用いてその結果に応じて情報を活用する．生成された質問は対話システムにより，理解度に合わせて段階的に質問を提示する．理解度に合わせるため，対話システムのはじめに，理解度状況を把握するための質問を行う．

2　関連研究

武藤らは模範解答を基にチャットボットが質問する学習支援方法を提案している [1]．模範解答を使用し，学習者が回答しやすく効率の良い質問木を生成する．質問木とは，質問同士のノードの繋がりを表した木構造である．模範解答から質問を生成することで，問題ごとに特化した質問を行うことを実現している．

松井らはプログラミング行動中の思考分析に基づいた初学者支援システムを提案している [2]．学習者とシステムの間で双方向の情報交換をし，決まった質問に決まった答えを返すという簡易的な質問と回答を繰り返す．指導者と学習者の対話のシステム化により自立的な学習を促す．質問はあらかじめ設定された特定の文字列が入力されることで提示される．例えば「for」と入力されるとあらかじめ設定されていた特定の文字列が表示される．

松浦は日本語教育における教師の学習者に対する効果的な訂正方法の分析をしている [3]．松浦は日本語教育にて見られる文法の誤りに対するフィードバックに着目した．効果的な訂正フィードバックにより，学習者はわからなかったことがわかったという満足につながるので，効率的かつわかりやすい訂正フィードバックをする必要がある．

長らは学習者履歴を用いて，プログラミングの授業にまつわる問題点を解決する [4]．プログラミングの授業にまつわる問題点とは，実習環境に起因する困難と大きな能力差に基づく困難である．これらを学習履歴から解析し，類型化する．類型に対するアドバイスを用意し自動提示できるようにしている．類型に対するアドバイスは，学習者が起こした事例をもとにそのパターンを意味や原因とともに学習者に直接提示する．

3　対話式誤り修正支援システムの提案

3.1　対話式誤り修正支援システムの概要

本研究では学習者のソースコードと理解度をもとに対話により修正を支援するシステムを提案する．本システムは，C 言語初学者がプログラムを作成した際にコンパイルは成功するが期待した実行結果が得られない場合に使用される．生成された質問グラフを利用した対話により学習者自身で修正できることが期待される．質問グラフのノードはシステムからの質問と学習者の回答，辺はノード間の遷移とする．これらの機能をもとにしたシステムの概要図を図 1 に示す．誤り推定ツールは 1 節にて説明してある．解析結果は誤り推定ツールから取得した情報とする．グラフ生成ツールは，本研究で作成した質問グラフのひな型に解析結果と誤りに対する登録情報よりグラフが生成されるツールとする．誤りに対する登録情報は誤りの種類ごとに指導者が事前に登録する内容とする．

質問グラフのひな型は定型の提示文とする．質問グラフはひな型に「誤りに対する登録情報」の情報が入り，質問グラフが自動的に生成される．ひな型を段階的な質問にし，学習者に有効性のある質問にする必要がある．質問グラフの質問の役割を「理解度状況把握の質問グラフ」と「誤りに対する質問グラフ」に分ける．「理解度状況把握の質問グ

ラフ」があることで冗長な質問をしないようにする．「誤りに対する質問グラフ」の役割として「誤り特定の質問」と「誤り修正の質問」に分ける．「誤り特定の質問」では学習者に誤り箇所を認識させる質問をし，「誤り修正の質問」では具体的なヒントを提示して修正を行える質問をする．

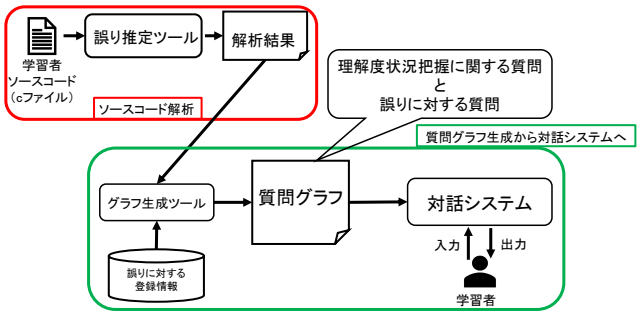

図 1　システム全体の流れ

本研究における対話システムの利点を 3 つ挙げる．

1. 段階的な質問を提示することで誤り箇所を自身で発見できるようにする
2. 対話形式により親しみやすく利用しやすい
3. 対話システムから質問文を提示し，学習者には選択肢や行番号など回答しやすい入力を用いる

3.2　誤りの分析

3.2.1　対象とする誤り

本研究では，潜在的な誤りを対象とする．その誤りの例として未初期化，配列の範囲外の添字の参照，無限ループ，条件式の多重が挙げられる．これらの誤りは，コンパイル時にエラーメッセージが出力されないことから修正できない要因となる．初学者はコンパイラのオプションについての知識がない，警告メッセージを表示しても意味を理解することができない可能性がある．

具体例として，ソースコード 1 のような条件式の多重やソースコード 2 のような配列の初期化または配列の範囲外の参照が考えられる誤りが挙げられる．

ソースコード 1　誤り例の一部：条件式の多重

```
5     int a,b,c;
6
7     printf("２つの整数を入力してください\n");
8     printf("整数a:"); scanf("%d", &a);
9     printf("整数b:"); scanf("%d", &b);
10
11    c = a - b;
12
13    if(-10 <= c || c <= 10)
14        printf("それらの差は 10以下です\n");
15    else
16        printf("それらの差は 11以上です\n");
```

ソースコード 2　誤り例の一部：配列の初期化または範囲外の参照

```
5     int numbers[5] = {1, 2, 3, 4, 5};
6
7     printf("%d\n", numbers[5]);
```

3.2.2　誤りの検出方法

本研究では誤り検出ツールとして cc -Wall，構文解析器として TEBA[5] を使用する．構文解析器では，ソースコードから関数や制御文を抽出する．この情報は質問グラフの生成に利用する．

3.2.3　誤りの分類

本研究ではシステムの過程で誤りの種類を特定型，非特定型に分類する．特定型は，誤り推定ツールで検出される誤りの行番号と修正すべき箇所が一致する誤りである．非特定型は，誤り推定ツールで検出される誤りの行番号と修正箇所が異なる誤りである．

「配列」のように誤りの種類ごとに分類することに加え，誤り修正箇所の行番号が一意に定まるかでも分類する．システム側で行番号の特定ができるので特定型と非特定型に分ける．

ソースコード 1 は，13 行目の重複した範囲に対して || を用いているので，多重の誤りである．図 2 により検出された誤りと修正すべき箇所の行番号が一致しているので特定型になる．

ソースコード 2 の誤りでは，7 行目にて範囲外の参照をしているので誤りが検出される．図 3 より誤りが検出されることがわかるが，このソースコードの修正すべき箇所は 7 行目とは限らない．5 行目の宣言にて配列のサイズを増やすことでも修正は可能であるので非特定型になる．

本研究では，cc -Wall によって検出される 13 種類の誤りを特定型 8 種類，非特定型 5 種類に分類して扱う．

3.3　質問の分析

3.3.1　質問の分類

本研究では質問グラフの前半を「理解度状況把握の質問グラフ」，後半を「誤りに対する質問グラフ」とする．質問グラフを分類することには，学習者に対応した質問を提示する意図がある．

学習者自身が誤りに気づき修正するには，誤りの種類や学習者の理解度に対応した質問にする必要ある．

理解度状況把握の質問グラフは，学習者のソースコードの解析結果から生成する．質問グラフは，全ての誤りに共通して使用するひな型とソースコードから抽出した関数名や制御文により構成される．検出される誤りの種類が特定型か非特定型かによって質問内容が変わる．

誤りに対する質問グラフは，学習者のソースコードの解析結果に加えて学習者の理解度から生成する．理解度状況把握の質問と同様，事前に用意しておいたひな型と，ソー

```
Example1.c:13:17: warning: overlapping
comparisons always evaluate to true
[-Wtautological-overlap-compare]
    if(-10 <= c || c <= 10)
       ~~~~~~~~~~~~~~~~~~~
1 warning generated.
```

図 2 cc -Wall のメッセージ 1

```
Example2.c:7:20: warning: array index 5 is
past the end of the array (Which contains
5 elemants) [-Warray-bounds]
    printf("%d\n", numbers[5]);
                   ^        ~
Example2.c:5:5: note: array 'numbers'
declared here
    int numbers[5] = {1, 2, 3, 4, 5};
    ^
1 warning generated.
```

図 3 cc -Wall のメッセージ 2

スコードから抽出した関数名や制御文を使用する．

誤りの種類ごとに誤りに対する登録情報を登録することで，より具体的なヒントを提示できる．

3.3.2 理解度状況把握の質問グラフ

3.2.3 節の分類を用いて「理解度状況把握の質問」を行う．

想定される学習者の状況は，誤り箇所がわかって修正できる・誤り箇所がわかるが修正できない・誤り箇所がわからなくて修正もできない・誤り箇所がわからないが修正できるの 4 通りである．誤り箇所がわかって修正できる人はこのシステムを利用する必要がない．また，誤り箇所がわからないが修正できる人はあり得ないので本研究では学習者の理解度を次の 2 種類に分類する．

- レベル 1：誤り箇所がわかるが修正できない人
- レベル 2：誤り箇所がわからなくて修正もできない人

レベル 1，2 のように誤り箇所の理解から分類するので 3.2.3 節の特定型・非特定型のように行番号で判断する．レベルを分類することにより，誤りの種類によって必要な質問を追加することができ，技術的課題の「段階的なヒントの提示」が可能になる．

学習者の状況をレベルごとに分類するために，特定型の質問グラフのひな型を図 4 に示す．図 4 において，ノードは 4 種類，辺は 2 種類ある．システムからの提示文ノード，四角形は学習者の入力・選択ノード，六角形は学習者のレベルに流すノード，破線の楕円は質問されないノードをそれぞれ表す．実線はシステム上の遷移，破線は学習者の入力・選択をそれぞれ表す．非特定型の場合は，行番号の質問と入力のノードを除いた質問グラフになる．

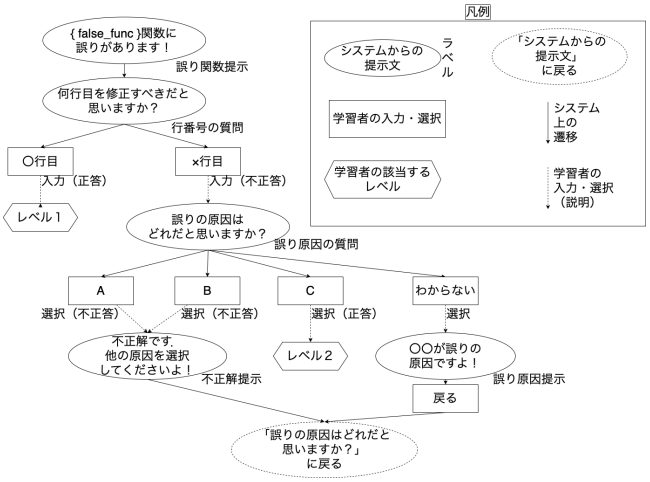

図 4 特定型の理解度状況把握質問グラフのひな型

3.3.3 誤りに対する質問グラフ

検出される警告メッセージごとに図 1 の誤りに対する登録情報を保存する．この登録情報は質問グラフのひな型のパラメータとして適用される．

質問グラフの大まかな流れを以下に示す．

1. 誤りのある関数名の提示
2. 行番号の入力（特定型の場合のみ）
3. 誤り原因の選択
4. ヒント提示

誤り原因の選択後，提示するヒントは 6 個ある．前半では具体的な情報を提示せずに学習者自身に考える余地を残し，後半では基礎知識や実際の誤り例などを用いて誤りの理解を深めて修正できるようにする意図がある．その内容を以下に示す．

1. 処理に関する質問
2. 誤りに対してのアプローチに関する質問
3. デバッグ方法に関する質問
4. よくある誤りを用いた質問
5. 基礎知識と例を用いた質問
6. 検索ワードを用いた質問

これらの提示する 6 個のヒントは段階的な質問にするためにを利用している．

図 5 の部分は質問グラフの「よくある誤りを用いた質問」の箇所についてである．この例はソースコード 2 の誤りの際に提示される文になっている．学習者に「いいえ」が選択された場合，さらに「基礎知識と例を用いた質問」が提示されていく．

これらのように図 5 の「いいえ」が押されることで提示文が進む．また，「はい」が押されると提示文が終了する．

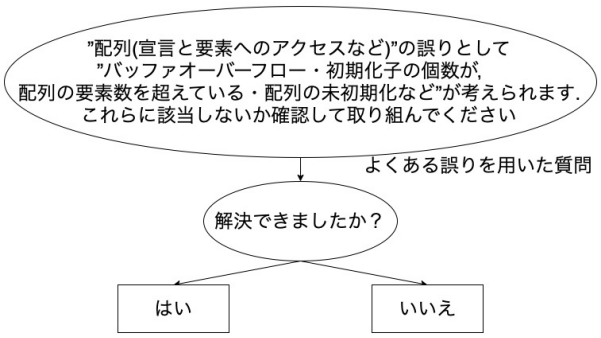

図 5 誤りに対する質問の一部分

4 評価

本システムは，学習者に誤りを認識させる質問を提示できているか，提示文によって学習者は誤りを修正できたか評価する．

4.1 対象・条件

ソフトウェア工学を専攻している大学 3 年生 4 人を対象に，潜在的な誤りのあるソースコードの修正をする検証を行った．4 人をそれぞれ被験者 A，B，C，D とし， 被験者 A，B はシステム使用有り，被験者 C，D はシステム使用無しで行った．

被験者には，問題文，誤りを含んだソースコードと正しい実行例を見せ，1 問ごとの制限時間を 10 分としてソースコードを修正してもらう．ソースコードの編集は可能であり，コンパイルや実行の回数の制限はなしとする．答えや解き方に関する質問，アドバイス，周りとの相談は不可とした．その様子の画面記録から，それぞれの学習者の修正方法や費やした時間から評価した．演習問題として，それぞれ違う誤りの C 言語の問題を 4 問用意した．

4.2 結果

被験者 A，B と被験者 C，D を比較すると，正しく修正できた数がそれぞれ 8 問中 7 問，8 問中 5 問とシステムを使用した被験者 A，B の方が修正成功している割合が多かった．また，3.3.3 節の 6 個のヒントに被験者 A，B が費やした時間を比較すると，2，3 は時間が少なく，4 に時間を多く割いていた．この結果より，2，3 は被験者に有益でない提示文になっているといえる．このヒントは，ログやフローを確認させたり，デバッグ作業を促したりと実際に被験者が手を動かして誤りの箇所や原因について考える必要があるので，考えるのをやめて次のヒントに進んでいる可能性もある．被験者は誤りに直接的に関連する情報を求めているといえる．4 のヒントは被験者に考える余地を与えて正しく修正に促すことができたといえる．

今回の評価は評価対象が少ないことや被験者のプログラミング能力のレベルを事前に把握できていなかったので，修正の成否や本システムの提示文の有用性の一般性については改めて評価する必要がある．

5 考察

5.1 潜在的な誤りが 1 つの関数に複数ある場合

本研究では，最初に表示された警告メッセージのみに対してヒントを提示し，修正できるようにアプローチする．警告メッセージが複数ある場合一つの誤りを解決すれば他の警告メッセージが解消される場合がある．本研究の方法では学習者との認識のズレが生じる可能性があるが最初に表示された警告メッセージを解決することで素早くプログラム全体を修正できる．どちらにも利点があるが本研究では最初に表示された警告メッセージのみに対してアプローチしていく方法をとる．

5.2 拡張性

本研究では，誤りの検出方法として cc -Wall を用いて 13 種類の誤りを扱うことができる．本研究で必要な誤りの情報は，行番号と誤りに対してのメッセージである．この 2 つの情報がある場合，行番号の一致確認が行え，質問グラフに利用する情報を登録することも可能になる．cc -Wall 以外の cppcheck や splint など外部ツールを利用することで，対応できる誤りの範囲が広がる．

6 おわりに

本研究では，プログラムの実行結果が学習者が期待したものと異なる場合に，対話により学習者の意図するコーディングができるように促すシステムを提案した．対応できる誤り数を増やしたり学習者の理解度や誤りの分類の精度の向上，多くの学習者を対象にした評価などが今後の課題である．

参考文献

[1] 武藤健太，西山雄也： チャットボットを利用したプログラミング学習者に対するコーディング支援方法の提案 —模範解答プログラムを利用した質問木の生成— ，2021 年度南山大学卒業論文 (2022).

[2] 松井大成，伊藤恵： プログラミング行動中の思考分析に基づいた初学者支援システムの提案, 日本ソフトウェア科学会第 35 回大会 (2018 年度) 講演論文集 (2018).

[3] 松浦沙樹： 教師の訂正フィードバックの分析，国際教養大学専門職大学院グローバル・コミュニケーション実践研究科日本語教育実践領域実習報告論文集, Vol.1, No.0，pp.69-93 (2010).

[4] 長慎也，筧捷彦： proGrep - プログラミング学習履歴検索システム，情報処理学会研究報告，Vol.2005-CE, No.15，pp.29-36 (2005).

[5] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満： 属性付き字句系列に基づくソースコード書き換え支援環境, 情報処理学会論文誌，Vol.53，No.7，pp.1832-1849 (2012).