教育意図を利用したプログラムのプルーフリーダの提案

2012SE027 後藤悠太　2012SE172 長谷優磨　2012SE246 田原寛隆
指導教員：蜂巣吉成

1　はじめに

大学でプログラミング演習を行う際，学習者は教育者から課題を与えられ，プログラムを作成するといった形式が多い．課題には実行例が載っているので，学習者は実行例通りにプログラムが動作すれば良いと考える傾向があり，教育者の意図と異なるコードを書いている学習者もいる．このような場合，教育者が学習者に対して意図と異なる記述を指摘する方法で解決できるが，教育者が学習者一人ひとりに対して指導する必要があり，負担が大きい．教育者が学習者に対して模範解答を開示し，学習者が自身の解答と比較するという方法もあるが，学習者は内部動作を考えないことが多く，どこが悪いのか分からない場合がある．

2014 年度の卒業研究 “ プログラミング演習における模範解答派生機能を備えた学習用校正ツールの提案 ” [1] では，学習者の解答に教育意図を含んだ記述があるか判定するツールの提案をしている．この研究では，模範解答から教育意図が記述されている箇所を教育意図パターンとして抽出する．その教育意図パターンが学習者の解答に記述されているかを調べ，判定を行っている．しかし，すべての課題において教育意図パターンが一律のルールで抽出されているので，教育者が意図しない箇所が抽出される場合や抽出して欲しい箇所が抽出されない問題がある．これにより，適切な学習者の解答が不適切と判定されたり，不適切と判定したい解答を適切と判定することがある．

本研究では，課題ごとに学ばせたい記述が異なると考え，学習内容ごとに異なる抽出基準で教育者の模範解答から教育意図を満たした箇所を抽出することで，2014 年度研究の問題点を解決する．本研究では，課題ごとに教育者が学習者に学ばせたいことを教育意図とした．学習項目は，明解 C 言語入門編 [6] の章立てを参考に，学習内容ごとにまとめたものである．教育者が課題ごとに模範解答を用意し，学習項目を選択することで模範解答から異なる基準で教育意図を含んだ記述を抽出することができる．学習者の解答からも同じ基準で教育意図を含んだ記述を抽出し，それぞれ抽出されたコード断片を比較することで不適切な学習者の解答を検出することができる．本研究で提案するプルーフリーダによって，学習者の学習効率向上と教育者の負担軽減が期待される．

本研究で提案するツールは，プログラミング演習で学習者がテストを行った上で教育意図と合っているかを確認するために使用されることを想定している．したがって，学習者のコードは「コンパイルに成功し，実行結果が正しいプログラム」を対象とする．

2　関連研究

コーディングチェッカや学習者向きのフィードバックを行うツールはいくつか提案されている．CX-Checker[2] は，C 言語プログラムのコーディングチェッカである．コーディング規約に基づき静的解析を行う．また，XPath などを用いてルールをカスタマイズすることができる．課題ごとに教育者の意図を考慮した指摘やアドバイスを与えるためには，教育者が課題ごとにルールをカスタマイズする必要があり，負担が大きい．proGrep[3] は，プログラムの翻訳 (コンパイル)・実行と同時に履歴サーバに学習履歴を蓄積し，学習履歴を利用して自動でパターンに記述されたアドバイスを行う．コンパイラの出力に対してアドバイスを行うので，作成したプログラムが教育者の意図を満たしているかどうかは判定できない．C-Helper[4] は，静的解析を用いてあらかじめ決められたルールに基づき，学習者が陥りやすいミスを発見，指摘し，可能な範囲で解決策を提案する．C-Helper[4] で決められたルールで指摘するので，課題ごとに異なる指摘を行うことはできない．

3　教育意図を利用したプルーフリーダ

本研究では，模範解答を用いることにより一般的なコーディングチェッカではチェックすることが困難な課題に対して正しく判定する．学習者が学習内容ごとの教育意図を満たした記述をしているかをチェックするために，学習内容ごとに異なる抽出基準で教育者の模範解答と学習者の解答から教育意図が記されているコード断片を抽出する．抽出したコード断片はそれぞれマッチングさせるために変数名や関数名の抽象化を行う．これを評価コードとする．2 つの評価コードをマッチングすることで判定を行う．想定される課題としてソースコード 1 の模範解答に対しソースコード 2，ソースコード 5 の模範解答に対しソースコード 6 のような学習者の解答が考えられる．

ソースコード 1　模範解答のコード断片

```
1 max = array[0];
2 for( i = 1 ; i < size ; i++){
3     if(max < array[i]) max = array[i];
4 }
```

ソースコード 2　学習者の解答のコード断片

```
1 max = 0;
2 for( i = 0 ; i < size ; i++){
3     if(max < array[i]) max = array[i];
4 }
```

ソースコード 3 ソースコード 1 の評価コード

```
1 =[]
2 for(=1;<;++){
3 $var=
4 }
```

ソースコード 4 ソースコード 2 の評価コード

```
1 =0
2 for(=0;<;++){
3 $var=
4 }
```

ソースコード 1 は，「配列の最大値を求めるプログラムを作成しなさい」という課題に対する模範解答のコード断片で，教育意図は「配列の先頭要素の処理を行って，残りの要素を順に走査し処理を行う」である．ソースコード 1 で示したコード断片は要素数 size(≥1) の配列 array から最大値を求める処理であり，まず配列の先頭要素を仮に最大とし，2 番目の要素から for 文で順に走査するのが適切である．これに対しソースコード 2 の学習者の解答では，変数 max は 0 で初期化され，for 文で配列の先頭要素から順に走査を行っている．この書き方は定数 0 を最大値として仮定しており，教育意図と合わない解答と考える．この場合評価コードはソースコード 3，4 のようになり，それぞれ比べると 1，2 行目が異なるので，学習者のコードは不適切であると判断される．

ソースコード 5 模範解答のコード断片

```
1 sum = 0;
2 for( i = 0 ; i < size ; i++ ){
3     sum += array[i];
4 }
```

ソースコード 6 学習者の解答のコード断片

```
1 sum = array[0];
2 for( i = 1 ; i < size ; i++ ){
3     sum += array[i];
4 }
```

ソースコード 7 ソースコード 5 の評価コード

```
1 =0;
2 for(=0;<;++){
3 $var=
4 }
```

ソースコード 8 ソースコード 6 の評価コード

```
1 =[]
2 for(=1;<;++){
3 $var=
4 }
```

ソースコード 5 は，「配列の合計値を求めるプログラムを作成しなさい」という課題に対する模範解答のコード断片で，教育意図は「合計を例に，配列の先頭要素から最後まで順に走査し処理を行う」である．ソースコード 5 は要素数 (≥0) の配列の合計値を求める処理であり，この場合は for 文のカウンタ変数は 0 から繰り返し処理させることが適切である．これに対しソースコード 6 の学習者の解答は，変数 sum を配列の先頭で初期化しており，配列の途中から繰り返しの処理をしている．このような解答も教育意図と合わないと考えられる．また，評価コードはソースコード 7，8 のようになりそれぞれ比べると 1，2 行目が異なるので，学習者のコードは不適切であると判断される．

最大値を求めるには for 文は 1 から繰り返し処理をするほうが良いが，合計を求めるには for 文は 0 から繰り返し処理をする方が良い．しかし, 一般的なコーディングチェッカではソースコード 1，6 のコードを区別して判定することができない．模範解答から評価コードの抽出を行うことによりこのような問題に対処できると考えた．

次に「a^n を計算するプログラムを作成しなさい」という課題を例に考える．この課題の教育意図は「for 文で n 回繰り返し計算を行う」である．模範解答は変数 pow を 1 で初期化し，for 文で i=0 から i<n まで繰り返し pow に a を掛けて計算する．この課題に対して，pow を 1 で初期化し，for 文で i=1 から i<=n まで繰り返し pow に a を掛けて計算する解答や，pow を a で初期化し，for 文で i=1 から i<n まで繰り返し pow に a を掛けて計算する解答が想定される．教育意図から前者の解答は適切，後者の解答は不適切である．しかし，昨年度の研究 [1] では抽出基準を一律としているので前者の解答も不適切であると判定されてしまう．正しく判定を行うためには課題ごとに異なる基準で評価コードを抽出する必要がある．

4 学習項目と評価コード

4.1 学習項目の必要性

本研究では，学習内容ごとに異なる抽出基準で模範解答から評価コードを抽出することにより，昨年度研究の問題点を解決する．抽出基準を課題ごとに教育者が記述することでこの問題は解決できるが，この方法では教育者の負担が大きい．そこで，学習項目ごとの評価コードに同じ特徴が現れれば，学習項目に対し抽出基準を設定することで，教育者が学習項目を選択するだけでその課題に適した抽出基準を用いて抽出できると考えた．これにより，教育者の負担は少なく，評価コードには過不足ない教育意図を表した記述のみを残すことができる．そこで，我々はどのような学習項目があり，それに対する評価コードはどのような形で出現するのかを調べた．

4.2 学習項目の決定

学習項目を決定するに当たっては明解 C 言語入門編 [6] の章立てを参考にした．その結果，学習項目は条件分

岐 (if)，条件分岐 (switch)，繰り返し (while)，繰り返し (for)，繰り返し (do-while)，配列の 6 つに分類できた．また，同文献のサンプルコードから学習項目ごとに評価コードを抜き出した．なお，学習項目の決定において評価コードを抜き出す作業は手作業で行った．その結果評価コードを学習項目ごとに分類し，抽象化することができた．これを意図パターンとする．

4.3 評価コードと意図パターン

意図パターンとは，評価コードを自動で抽出するために使用するパターンであり，学習項目ごとの評価コードの特徴的な箇所を示している．意図パターンは学習項目と条件式，初期化の有無によって記述が変わり，条件式，初期化の有無は教育者が判断する．教育者は学習項目を選択することで，意図パターンが決まり，この意図パターンを用いて模範解答から自動で評価コードを抽出することができる．3 節であげた「配列の最大値を求める」課題と「配列の合計値を求める」課題を例に考える．

ソースコード 9 意図パターン (配列初期化有り)

```
1 ={ 値又は [] }
2 for(=値;関係演算子;演算子){
3 $var=
4 }
```

ソースコード 10 意図パターン (for 文条件式無し初期化有り)

```
1 =値
2 for(){
3 $var=
4 }
```

ソースコード 11 a^n を計算するプログラムの評価コード

```
1 =1
2 for(){
3 $var=
4 }
```

この 2 つの課題の学習項目は，教育意図から「配列 (初期化有り)」となる．配列は条件式を必ず抜き出すので条件式の有無は問わない．この学習項目に対して定義されている意図パターンはソースコード 9 になる．意図パターン中の**値**，**関係演算子**，**演算子**はソースコード中の該当する箇所をそのまま評価コードに抽出する．[] はソースコードの該当する箇所に配列が使われていれば [] を抽出する．$var はソースコード中の変数を$var に変換した記述である．ソースコード 1，5 にソースコード 9 をパターンマッチングさせることで，ソースコード 3，7 が抽出される．また，3 節であげた「a^n を計算するプログラムを作成しなさい」という課題を例に考える．教育意図から学習項目は「for 文 (条件式無し初期化有り)」となり，この学習項目で定義されている意図パターンはソースコード 10 となる．模範解答にソースコード 10 をパターンマッチングさせることで，評価コード 11 が抽出される．この評価コードを用いることにより，学習者の解答に，変数 pow を 1 で初期化する初期化式があるか，for 文が記述されているか調べることができ，3 節であげた学習者の解答を正しく判定することができる．学習項目を選択することで，意図パターンが決まり，課題ごとに異なる抽出基準で評価コードが抽出できる．

5 ツールの実現

本研究では，教育者が利用する「教育意図抽出ツール」と学習者が利用する「教育意図判定ツール」を実現した．プルーフリーダの全体像を図 1 に示す．

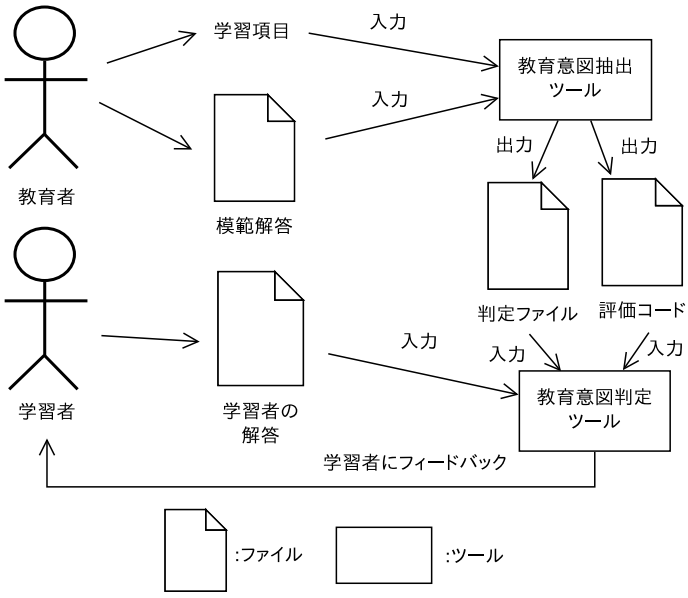

図 1 ツールの全体像

教育者は学習者が教育意図判定ツールを利用する前に，作成した模範解答のファイル名，その課題に対する学習項目を教育意図抽出ツールに入力する．入力された学習項目に対して意図パターンが定義されているので，その意図パターンと模範解答を用いて評価コードと判定ファイルを出力する．判定ファイルには，入力された学習項目と評価コードのファイル名が記述されている．その後，学習者は教育者から判定ファイルと評価コードを受け取り，教育意図判定ツールに解答のファイル名を入力する．これにより，学習者の解答から評価コードが抽出され，模範解答の評価コードとマッチングを行い，学習者にフィードバックする．ツールの実装には，パターンに基づいたコード変換が可能である TEBA[5] を利用した．

6 検証

南山大学情報理工学部 1 年生を対象に開講された 2012 年度秋学期プログラミング基礎実習の課題（計 59 問）と

これらの課題に解答をした 9 人の学生の解答を対象に，提案したツールを用いて不適切な学習者の解答が検出できるか確認を行った．対象とした課題 59 問中 19 問は，異なる解答を記述している学習者はいなかったので，ツールを利用する必要がない課題として考え，集計の対象外とした．また模範解答の中には，設定した学習項目が含まれない課題も存在したので，ツールの検証には，残りの 36 問を対象に集計を行った．

ツールを利用したところ 36 問中 22 問で不適切な学習者の解答が検出された．検出された課題として，課題内容「3 つの整数を入力として受け取り，値が等しい数を出力するプログラムを作成しなさい」，教育意図「if 文と else 文を用いて適切な条件式の記述」という課題がある．模範解答のコード断片はソースコード 12 となる．この課題に対して入力された 3 つの整数がすべて等しい場合の条件式をソースコード 13 のように記述している学習者の解答があった. この学習者の解答には, 不要な条件式の記述があるので，この解答は不適切であるという出力を得られた．

ソースコード 12 模範解答のコード断片

```
1 if (na == nb && na == nc)
2     printf("三つの値は等しいです。");
```

ソースコード 13 学習者の解答のコード断片

```
1 if (na == nb && na == nc && nb == nc)
2     printf("三つの値は等しいです。");
```

また判定ができなかった課題は 14 問あり，大きく分けて 2 種類に分類することができた．1 つ目は，複数の学習項目がある課題である．これは，本ツールが複数の学習項目の入力に対応していなかったので，評価コードの抽出ができなかった．この問題に対して，7.1 節で考察する．2 つ目は，適切な学習者の解答を不適切と判定してしまう課題である．これは，模範解答が複数記述できることが原因である．この問題は，教育者が複数の模範解答を用意し，それらの模範解答を用いて学習者の解答を判定することで，解決することができる．

7 考察

7.1 複合課題

繰り返しと条件分岐の入れ子構造のように 1 つの課題に学習項目が複数存在することがある．現在の実装では，複数の学習項目を用いて，一度に評価コードを抽出することができない．この問題に対しては，ツールを 2 回用いてそれぞれの学習項目に対して評価コードを抽出する方法で解決することができる．しかし，この方法ではソースコードの構造まで確認することができない．検証では，正しく判定できない課題は見つかっていないが，構造が変化しても正しく動作し，構造が重要な課題があるかもしれないので，更なる検証が必要である．

7.2 アンチパターン

課題には，模範解答には現れない記述をしてはいけないという課題もある．このような課題にも対応するために，アンチパターンを利用する．アンチパターンとは，学習者の解答に不適切な記述がないかを調べるために利用する意図パターンである．具体的には,「配列の合計と平均を求めるプログラムを作成しなさい」という課題内容で，合計を求める for 文の中で平均を求める処理をしている解答などが想定される．この解答に対して,「for 文の中に”/” を用いた計算がある」というアンチパターンを適用させると，不適切な記述なので，模範解答にはマッチングせず，不適切な学習者の解答のみにマッチングする．よって，不適切な解答から抽出した評価コードと模範解答から抽出した評価コードとで違いが出るので，このような解答を不適切であるという判定を行うことができる．

8 おわりに

本研究では，学習者が教育意図を満たした解答をすることを目的に，模範解答と学習者の解答から教育意図にあたる箇所を評価コードとして抽出し，それらを照らし合わせフィードバックを行う教育意図を考慮したプルーフリーダを提案した．今後の課題は，学習者に対するフィードバック方法の決定，複合課題の解決である．

9 参考文献

[1] 堀尾美貴，金崎真奈美，佐藤成, “ プログラミング演習における模範解答派生機能を備えた学習用校正ツールの提案 ”，南山大学情報理工学部 2014 年度卒業論文, 2014.

[2] 大須賀俊憲, 小林隆志, et al., “ CX-Checker：柔軟なカスタマイズが可能な C 言語コーディングルールチェッカー ”，情報処理学会論文誌， Vol. 53， No.2，pp. 590-600， 2012.

[3] 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．コンピュータと教育研究会報告，vol. 2005，NO. 15，pp. 29-36，2005.

[4] 内田公太，権藤克彦, “ C-Helper:C 言語初学習者向け静的解析ツールの提案 ”，ソフトウェア工学の基礎 XIX 日本ソフトウェア科学会 FOSE2012，近代科学社，pp. 231-232， 2012.

[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.

[6] 柴田望洋，“[新版] 明解 C 言語入門編”, ソフトバンククリエイティブ株式会社, 東京, 2004.