K-034

プログラミング演習における学習者の到達度を考慮した小テスト生成

Generation of Examination to Evaluate Understanding Levels in Programming Class

佐藤 直崇 † **原田 史子** ‡ **島川 博光** ‡
Naotaka Sato Fumiko Harada Hiromitsu Shimakawa

1. はじめに

多くの大学では学生がコンピュータの仕組みの構造的な理解を深めるためにプログラミング演習が行われている．プログラミング演習とは講義で学んだ知識を活用し，ソースコードを作成する演習である．教員は学習者のプログラミング技術の到達度を把握することにより，学習者の成績を評価している．プログラミングの到達度を把握するには定期試験やソースコードの評価が実施されている．しかし定期試験は実施回数が少なく出題範囲が広い．初めてプログラミングを学ぶ学習者を定期試験で評価することは難しい．一方，ソースコードの評価では学習者の到達度を詳細に評価できるが，教員や TA が学習者のソースコードを評価するための負荷が大きい．

本論文では学習者の到達度に合わせた小テストの生成を提案する．本手法では過去の学習者がプログラミングを習得するために作成したソースコードから課題の特徴を抽出する．この特徴から生成した小テストを学習者が解答し，その結果を考慮した小テストを生成することにより学習者の詳細な到達度を把握する．

2. 到達度の把握

教員は学習者のプログラミング技術の到達度を把握するためにソースコードの評価や定期試験を実施している．ソースコードの評価とは各週に与えられた課題を学習者がソースコードを作成することにより解く．作成されたソースコードを教員や TA が題意を満たしているかをチェックすることにより学習者の到達度を評価するものである．ソースコードの評価は複数の課題を多様な観点から評価することと，学習者のプログラミングの到達度を把握する機会が増すため，学習者の詳細な到達度を把握することに適している．しかし少数の教員・TA が多くの学習者のソースコードを評価することになるため，その負担は大きなものとなる．また定期試験による評価では，教員が学習者を評価する回数が少ないため負担が小さい．しかし評価の回数が少ないため一度に評価する対象範囲が広くなり，把握できる到達度の粒度が荒くなる．把握する到達度の粒度が荒くなると，多数の学習者にとって難易度の低い試験が作成されたさいには，高得点を取る学習者が大多数を占めるため適切な評価が難しくなる．また難易度の高い試験が作成されたさいには，多数の学習者が正解を導けないため試験の得点が低い学習者が大多数を占め，学習者を評価することが困難になる．よって学習者のプログラミングの到達度を評価するために，到達度の高い学習者が高得点を取得し，到達度の低い学習者が低得点を取得するような，学習者を適切に評価できる試験を作成することが求められている．

†立命館大学大学院理工学研究科
‡立命館大学情報理工学部

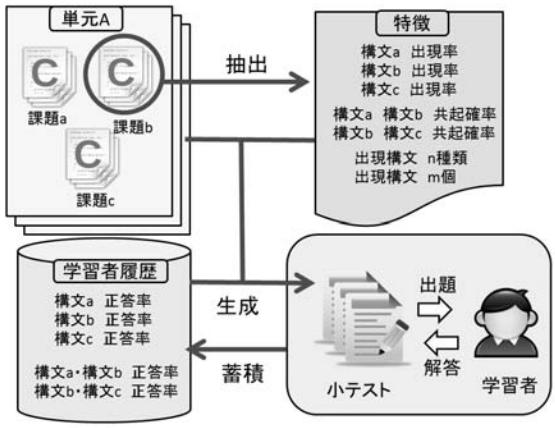

図 1: 手法の全体像

3. 学習者の到達度を考慮した小テスト生成

3.1 課題評価と到達度の考慮

本論文では学習者の到達度を詳細に把握し学習者を適切に評価するために，学習者の到達度を考慮した小テストの生成を提案する．図 1 に本手法の流れを示す．まず前年度の学習者が演習で課題を解くために作成したソースコードを評価する．これらのソースコードは 1 つの課題に対して多くの学習者が解いたものである．このソースコードの集合から各課題に対して，学習者が使用した構文の種類，構文の出現率，共に使用される構文といった特徴を抽出する．また新たにプログラミングを学ぶ学習者において，プログラミングに関する到達度を認識するために学習者履歴を取得する．学習者履歴とは学習者が今までに学んだ各構文の使い方や組合わせ方をどの程度理解しているかを蓄積したものである．そして学習者履歴に蓄積された学習者の到達度と各課題の特徴から，学習者を適切に評価するための小テストを生成する．これによって生成された小テストを学習者が解き，その結果を学習者履歴として蓄積する．このようにして学習者の到達度を考慮した小テストを生成する．

3.2 提出されたソースコード

本手法では単元ごとに過去の学習者が提出したソースコードを基に課題を評価する．プログラミング演習ではプログラミングを教える 1 つのまとまりとして，入出力，条件式と選択や繰返しといった単元がある．単元ごとに新しく学ぶ概念や構文があり，学習者は課題を解くことでこれらを習得する．単元ごとに課題は複数存在し，新しく学んだ構文の使い方のみを理解していれば解ける基本課題や，前単元で習得した概念や構文を組合わせて解く応用課題があり，単元が進むにつれ今まで学んだことが重要となる．たとえば入力の単元で数値や文字を入力する概念とそれを実現するための手法を学び，次に条件式と選択の単元で if 文を学んだとする．このとき条件式

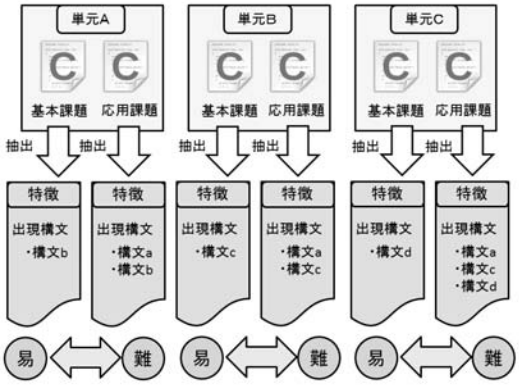

図 2: 特徴抽出

と選択の単元では，if 文を使用して解く基本課題だけでなく，入力された数値によって条件分岐をさせるような前単元で学んだ入力の概念を用いる応用課題がある．

提出されたソースコードは過去の学習者が課題を解いた成果物である．課題はソースコードを作成することで解かれるため，自由度が高く解法が複数通り存在し，学習者によって解き方が異なる．しかし各学習者が学ぶ単元の順番が同じであるため，各課題において多くの学習者が使用する構文の種類や順序に似通った特徴が現われ，これにより各課題を評価できると考えられる．

3.3 課題の特徴抽出

課題を解くために作成されるソースコードには多くの構文が使用される．構文の使用方法や構文の組合わせを学ぶことで学習者はプログラミングを習得する．そのためソースコード中の構文に着目することで課題の特徴を抽出する．図 2 にソースコードから課題の特徴を抽出する流れを示す．

まず課題は単元ごとに複数個用意されており，新しい単元で学んだ構文のみで解ける基本課題や，前単元で学んだ構文と組合わせて解く応用問題がある．本研究では，毎年同じ課題を学習者が解くものと仮定する．各課題に対して過去の学習者が解いたソースコードが複数存在するため，各ソースコードに使用されている構文を抽出する．ソースコードから抽出された構文の集まりから，より多くの学習者によって使用された構文の種類や構文の組合わせをその課題の特徴とする．たとえば 単元 A，単元 B の順に学ぶとき，単元 B の基本課題のソースコードから構文を抽出したさいに，構文 b が最も多く使用されているとき，単元 B の基本課題の特徴を 構文 b とする．また 単元 B の応用課題からは 単元 A で学んだ 構文 a と 単元 B で学んだ 構文 b が共に多くの学習者によって使用されていたさいには，単元 B の応用課題の特徴を 構文 a と 構文 b とする．

基本課題は新しい構文の使い方を学ぶため，ソースコードに使用される構文が少ない傾向にある．また応用課題は前単元を踏まえて解くため，ソースコードに使用される構文の種類が多くなる傾向にある．よって使用されている構文の種類や数によって課題の難易度を評価できると考えられる．

3.4 到達度を考慮した小テスト

学習者が各単元で学ぶ構文に関する小テストにより学習者の到達度を把握する．そのために前単元における課題の特徴から作成された小テストを学習者が解答し，その正誤により各学習者の到達度を把握する．各学習者の到達度からその単元における全学習者の正答率を計算し，次単元の小テストを生成するための難易度の指標とする．たとえば 単元 n と 単元 $n+1$ の正答率から 単元 $n+2$ の小テストの難易度を考える．x を 単元 n と 単元 $n+1$ を合わせた設問数中の 単元 n に関する設問数とする．このとき (単元 n の正答率) $\times x +$ (単元 $n+1$ の正答率) $\times (1-x)$ の値を 50 に近づけるように 単元 n と 単元 $n+1$ に関する設問の割合を調整する．単元 n の正答率が高く，単元 $n+1$ の正答率が低いときには x の値を小さくすることにより，単元 n に関する設問数を少なくし，単元 $n+1$ の設問数を多くする．また 単元 n の正答率が低く，単元 $n+1$ の正答率が高いときには x の値を大きくすることより，単元 n に関する設問数を多くし，単元 $n+1$ の設問数を少なくする．これにより 単元 $n+2$ の小テストの難易度を調整する．

課題から抽出された特徴を用いて小テストは生成される．抽出された課題の特徴はソースコード中の構文の種類，構文の出現回数や構文の組合わせである．たとえば 構文 a と 構文 b が共に出現回数が多いという特徴があれば，過去の学習者が解いたソースコードから 構文 a と 構文 b に関する小テストを作成する．生成される小テストは完成されたソースコードから 構文 a と 構文 b を穴抜きにした問題や，構文 a の代わりに異なる構文を記述した正誤問題，さらに 構文 a の部分を穴抜きにし空欄に入る正しい構文を選択する選択問題などが考えられる．

4. 関連研究

これまでにも学習者の理解度を考慮した問題作成の研究がされている．文献 [1] では教員が教材から出題したい部分を選択することで問題が生成されるため，教員が問題を作成する負荷を減らせる．また出題した問題に学習者が正解した場合には，その学習者の理解度を上げ，不正解であった場合には理解度を下げる仕組みがあるため学習者の理解度に応じた問題を出題できる．しかし学習者の理解度に合わせた問題を出題するために学習者ごとに出題される問題が異なるため，教員が学習者を公平に評価する指標としては利用できない．本手法では教員が学習者を公平に評価するための小テストを生成する．

5. おわりに

本論文では，学習者の到達度を考慮した小テストを生成する手法を提案した．今後は本論文で述べた手法から小テストを生成し，学習者を適切に評価するための小テストが生成されるかを検証する予定である．

参考文献

[1] 菅沼明，峯恒憲，正代隆義，“ 学生の理解度と問題の難易度を動的に評価する練習問題自動生成システム ”，情報処理学会論文誌，vol.46，No.7，pp.1810-pp.1818，Jul.2005.