摂動に基づくC言語の文法知識習得支援システム
A System to Acquire Grammatical Knowledge of Language C based on Perturbation

工藤永貴† 橋浦弘明‡ 古宮誠一‡
Hisaki Kudo† Hiroaki Hashiura‡ Seiichi Komiya‡

† 芝浦工業大学　工学部
‡ 芝浦工業大学大学院　工学研究科
† Faculty of Engineering, Shibaura Institute of Technology
‡ Graduate School of Engineering, Shibaura Institute of Technology

要旨
プログラムの文法説明書に書かれたコーディングサンプルに，システム主導で変更を加える(これを摂動と呼ぶ)とともに，その変更に絡む質問を学習者に出題する．このような学習法を「摂動に基づく学習法」と呼ぶ．システム主導で変更を加えたコーディングサンプルを実行するだけの学習法よりも，広く深い文法知識が効率よく習得できる．本稿では摂動に基づく学習法を利用して，如何にしてプログラム文法を学ぶか，その例を示す.

1. はじめに

プログラムの文法を習得するには，文法説明書に書かれたコーディングサンプルをコンピュータに入力し，コンパイルするとともに，文法書に書かれたデータを流して，文法書に書かれた通りの結果が得られることを確認する．この動作を繰り返すことによって文法を学習する．これはプログラムの文法を習得するために一般的に採用されている学習法であるが，これではコーディングサンプルに書かれた内容以上の知識は習得できない.

また，学習者はそのサンプルコードがどのような文法事項を習得させようと意図されて作成されたか考えずに，機械的にソースコードを打ち込んでしまうことがある．そのような学習者はサンプルコードのどの部分が出力に関係しているかが分からないため，全く同じようなコードしか挙動を想像することができなくなってしまう．そのため，そのソースコードからどのような知識を得たのか，教授者が問いかけ．知識を定着させる必要がある．しかし，学習者と教授者の数の比率を考えると，現実的な方法ではない.

そこで，システム主導でコーディングサンプルに変更を加える（これを摂動と呼ぶ）とともに，その変更に絡む質問を学習者に出題する．これを繰り返すことによって学習する方法を「摂動に基づく学習法」と呼ぶ．システム主導でプログラムに変更を加えるので，コーディングサンプルを実行するだけの学習法よりも広く深い文法知識が効率よく習得できると思われる．また，システムが学習者に質問することにより，学習者にソースコードのどの部分がどのような挙動に関係しているかを考えさせることができる.

本稿では，摂動に基づく学習法に基づいて，いかにしてプログラム文法を学ぶか，その例を示す.

2. 文法説明書を使った学習の問題点

if 文を例にとり，学習者がどのような事柄を学習しなければいけないかを考える．if 文における最も基本的な学習項目として，if 文の中に記述された条件式が真であるならば，if 文に続く文を実行する，というものがある.

しかし，文法書には単純な例しか載っておらず，学習者が記述すると思われる失敗例についてはほとんど載っていない．学習者が記述する誤ったプログラムは，ある入力値の時は正常に動作するが，別の値を入力したときは正しく動作しないといったことが多い．そのため，プログラムの誤りに気付かず，誤った文法知識を学習してしまうことがある.

たとえば，プログラミングの初心者は以下のようなプログラムを記述することがある．

```
int score;
scnaf(“%d”,&score);
if(0 <= score <= 100)
{
    printf(“valid value”);
}
```

```
int score;
scnaf(“%d”,&score);
if(0 <= score && score <= 100);
{
    printf(“valid value”);
}
```

```
int score;
scnaf(“%d”,&score);
if(0 <= score && score <= 100)
    if(score >= 80)
        printf(“good score!”);
else
    printf(“invalid value”);
```

図 1 誤っているが正しい動作をすることもある例

図 1 左のソースコードでは条件式を数学の式のように記述している．このプログラムは，はじめに 0<=score の部分を評価し，次にその結果が 100 以下であるかどうかを評価するため，常に条件式が真になる．しかし，学習者が入力値を 0 から 100 の場合しか試していなければ，この誤りに気付くことはできない．そのため，別の入力値を入れたときに突然プログラムがおかしくなったと感じ，プログラムの別の個所が誤りではないかと考えてしまう．

図 1 中央のソースコードでは，if 文の条件式の後にセミコロンが入っている．その場合，条件式が真だった時に実行される文が何もしない空文となってしまう．よって，本来条件式が真の時のみに実行するはずの文が常に実行されてしまう．このソースコードも，図 1 左と同様に，学習者が入力値を 0 から 100 の場合しか試していなければ，この誤りに気付くことはできない．

また，図 1 右のソースコードは，else 節と結合する if 文についての認識があやまっている．この場合でも，入力値が 80 から 100 の間は正しい挙動をする．

このように，初心者の誤りは様々であるが，通常の文法書ではこのような誤りの例を網羅することができない．

3. 提案する手法

前節で述べたように，学習者に正しい文法知識を習得させるためには，大量のコーディングサンプルを用意し，それぞれに対し，複数の適切な入力値を与えたり，サンプル内の構文の位置を適当に変えたりした上で実行させる必要がある．しかし，これを人手で用意するのは困難である．そこで，用意したコーディングサンプルに対し，様々な入力値を与え，適当にコーディングサンプルを変化させるようなツールを使うことで，様々なプログラムにおける構文の挙動を学習させることを提案する．

学習者はツールを用いて，以下の手順を繰り返すことで文法事項を学習する．

1. システムは学習させたい文法事項を決定する
2. 学習させたい文法事項に合わせた問題セットを取り出し，入力値や定数の値を変化させ，学習者に出題する
3. 学習者は問題に回答し，解答を確認する
4. 問題セットを全て回答したならば，学習者に何が類推できたかを問いかけ，学習させたい文法事項についての解説を表示する
5. 1 に戻る

図 2 で示した，if 文の後にセミコロンをつけてはいけないということを類推し，学習させる例を以下に示す．

問題
以下のプログラムは変数 score の値が 0～100 の間ならば”valid value”と画面に出力する、という目的のもと作成されている。
このプログラムを実行するとどのような結果になるか答えよ。

プログラム

```
int main(){
   int score=   110;
   if(0 <= score && score <= 100);
   {
      printf(“valid value”);
   }
}
```

回答
・valid value と出力されない
・valid value と出力される
・コンパイルエラーとなる

回答

図 3 出題の例

```
//①　正解となる
if(0 <= score && score <= 100)
{
   printf(“valid value”);
}
```

```
//②　正解となる
if(0 <= score && score <= 100)
   printf(“valid value”);
```

```
//③　セミコロンが付いていて誤り
if(0 <= score && score <= 100);
{
   printf(“valid value”);
}
```

```
//④　セミコロンが付いていて誤り
if(0 <= score && score <= 100);
   printf(“valid value”);
```

図 4　if 文の後のセミコロンについての問題セット

変数 score の値については，0 から 100 の間の値，0 未満，100 より上の三つの場合を満たす値をランダムに生成し，プログラムごとに割り当てる．つまり，四つのソースコードそれぞれに対し，score の値を変えた三つの問題を出題することになる．

出題したソースコード
① セミコロンが付いてないブロック
② セミコロンが付いてない改行
③ セミコロンが付いているブロック
④ セミコロンが付いている改行

実行結果（○：正しい出力　×：誤った出力）

番号/入力値	-50	50	150
①	○	○	○
②	○	○	○
③	×	○	×
④	×	○	×

類推されること
;が付いていると正しく動作しない

文法事項の解説
if 文は条件式が真の時、次に続く一文を実行する。条件式の直後に;があると、空文として実行され、本来条件式が真の時のみ実行される文が、常に実行されてしまう。

図 5 解説の例

4. まとめ

プログラムの文法書には正しいコーディングサンプルが載っているが，誤った記述についてはほとんど載っていない．また，学習者が記述するプログラムは誤りが含まれている場合があるが，正しい入力値を与えると正常に動作する場合が多い．そのため，学習者が誤った記述をした場合，それを誤りと気付かないまま覚えてしまうことがある．

覚えた文法知識が誤りだと学習者に気付かせるには，大量のコーディングサンプルが必要である．しかし，コーディングサンプルの作成を人手で行うのは困難なため，ツールを用いて解決する．

ツールでは，ある一つの文法知識について，正しい記述と誤った記述を提示し，学習者にどのような動作をするか質問する．学習者が複数の記述に対して解答した後，それらの記述からどのような文法知識が得られたか質問する. その後, 答え合わせとしてツールが意図した文法知識についての解説を行う．

このような手順を, 学習者が繰り返し行うことによって, 学習者に正しい文法知識を習得させていく．

5. 今後の課題

ツールには，あらかじめ，誤りプログラムを大量に用意しなければいけない．初心者の誤りプログラム例として，AZUR[1]の利用ログを分析し，問題を作成することを考えている．

文法知識習得のための効果的な出題の在り方，例えば，(摂動による)出題を繰り返し，一連の出題が何を意味するのか，学習者に帰納推論させ，具体例の集合から，それらが成り立つ一般的な法則を気付かせるような出題方法を模索することが本研究の狙いである．

参考文献

[1]. T. Imaizumi, H. Hashiura, S. Matsuura, S. Komiya, "A Programming Learning Environment "AZUR": Visualizing Block Structures and Program Function Behavior,"Proc. of JCKBSE2010, pp.306-311, Kaunas, Lithunia, Aug. 25-27, 2010.