既習プログラミング言語の知識に起因する誤りの自動修正方法の提案
―制御文を対象として―

2017SE008 東 直希　2017SE048 三上 比呂　2017SE053 長野 滉大
指導教員：蜂巣吉成

1　はじめに

現在，様々なプログラミング言語が存在し，学習者も複数のプログラミング言語を学ぶことが想定される．学習者が第二第三の言語を学ぶ際に，既習言語の知識と学習している言語の内容を混同し，記述ミスにつながることが問題として挙げられる．

間辺ら [1] の調査によると，C 言語の学習経験を持つ高校生を対象に，Python の授業を行ったところ，次のように誤った記述をした生徒が多数いた．

- 「変数の宣言は必要ない」という教師の指示にも関わらず変数宣言を記述した．
- 反復で「for i in range(10)」と表現すべきところを，「for(i = 0; i <= 9; i++)」と C 言語の記述をした．
- 記述不要な「;」を print 文などの後ろに記述した．

トランスコンパイラなどを用いれば，コンパイル可能なプログラムの言語変換は可能であるが，編集途中のソースコードに適用するには，次の技術的課題がある．

課題 1 記述されたソースコードが何の言語で書かれているのかを判定する方法

課題 2 正しい文法で書かれているとは限らない記述に対しての解析・変換方法

Listing1 に想定される誤り例を示す．Listing1 では，C 言語のソースコードに C 言語と Python の文法が混在した for 文が記述されている．

Listing 1　課題 2 の例

```
1 int main(void) {
2     int num, i;
3     printf("number?:"); scanf("%d",&num);
4     for (i in range(num)) {
```

本研究では，制御文 1 行のコードを対象として，誤って他言語の文法に基づいた記述や複数言語の文法が混在した記述をした際に，どの言語で書かれていたかをメッセージで出力し，目的言語の正しい記述に修正する方法を提案する．制御文は，言語によって大きく異なる場合や，似ていても紛らわしい違いがあるので，対象とした．本研究では，分岐，反復を表す共通モデルを定義し，共通モデルを介した変換方法を提案する．技術的課題に対しては，誤りの対象を定め，部分的に解析していき，言語の情報と目的言語への変換に必要な要素を抽出し変換することで解決する．対象とする言語は，C 言語，PHP，JavaScript，Python，Ruby とする．

2　関連研究

関連研究として，トランスコンパイラ，UNICOEN [2]，TransCoder [3] が挙げられる．

トランスコンパイラは，ある言語のソースコードから，他言語の同等のソースコードに変換するツールである．C 言語のソースコードを JavaScript に変換する Emscripten [4] などがある．

UNICOEN は複数の言語に対応したソースコード処理フレームワークである．C 言語, Java, C#, Visual Basic, JavaScript，Python，Ruby の 7 種類の言語について，それぞれの文法の和集合を取り，統合コードモデルを設計しており，各言語のソースコードと統合コードオブジェクトとの相互変換が可能である．

TransCoder は，Java，C++，Python を対象とした言語変換ができ，読みやすく自然なソースコードへの変換ができる．また，各言語の複雑な記述パターンも読み取り他言語に変換することができる．

トランスコンパイラや UNICOEN は，正しく記述されたソースファイルを他言語に変換するのに使われるツールなので，複数の言語が混在していたり，誤った記述が含まれるソースファイルに対する変換には対応していない．TransCoder は，機械学習を用いており，変換の精度は学習データなどに依存する．

3　各プログラミング言語の調査

本章では，今回対象とする言語の基本的な文法の違いを調べ，本研究の対象を定める．

3.1　変数

言語によって，変数に接頭辞「$」をつけるものと，つけるとエラーになるものがある．本研究では，制御文の中で書かれる変数の接頭辞「$」の有無について，メッセージを出力することで記述の間違いを指摘する．

3.2　制御文

3.2.1　分岐

分岐には，if 文と switch 文が存在する．

if 文は，elseif 節と else 節により条件を追加して記述される．C 言語，JavaScript には elseif 節が存在せず，else 節と if 文の組み合わせで記述する．また，PHP では，else 節と if 文の組み合わせと「elseif」のどちらでも記述することができる．switch 文は，C 言語，PHP，JavaScript については同様の文法である．Ruby では switch 文と似た構造の case 文が存在する．Python には同等の文は存在し

ない．また，switch 文は if 文で代用もでき，一般的に使用頻度が低い．本研究では，if 文を対象とする．

3.2.2 反復

反復には，for 文，foreach 文，while 文，do-while 文が存在する．

for 文は回数が決まっている反復をする場合に，foreach 文は配列・リストに格納されている値を順次処理する反復の場合に，while 文は回数が決まっていない反復をする場合に，do-while 文は処理を一度実行した後に条件を調べ反復する場合に利用される．for 文の文法を表 1 に示す．JavaScript，Python，Ruby については，for 文を用いて PHP の foreach 文と同等の処理を表現できる．また，回数が決まっている反復に関しては，各言語においてよく使われる記述パターンをイディオムとして定める．決められた回数繰り返す反復のイディオムを表 2 に示す．do-while 文は，C 言語，PHP，JavaScript について，同様の文法であり，Ruby では文法が異なり，Python においては同等の文法が存在しない．do-while 文は，while 文で代用もでき，一般的に使用頻度が低い．本研究では，for 文，foreach 文，while 文を対象とする．

表 1 for 文

C 言語 PHP JavaScript	**for** ([*expr*]; [*expr*]; [*expr*]) *statement*
Ruby	**for** *var* **in** *expr* [**do**] *statement*+ **end**
Python	**for** *var* **in** *expr* **:** *statement*+
JavaScript	**for** (*var* **of** *expr*) *statement* **for** (*var* **in** *expr*) *statement*

表 2 決められた回数繰り返すイディオム

C 言語 PHP JavaScript	**for** (*var* = *initial* ; *var* < *end* ; *var*++)
Ruby	**for** *var* **in** *initial*...*end*
Python	**for** *var* **in range**(*initial*,*end*)**:**

3.2.3 例外処理

例外処理は，C 言語では例外処理の構文がないことや，各言語で処理の構造が違う部分があり，コード 1 行からの変換で整合性を取ることができないので，扱わない．

3.3 式

式は，各言語で演算子に違いがある．演算子は，各言語で優先度や役割が異なる場合があるので，単純な書き換えでは厳密に整合性をとることはできない．本研究では，論理演算子と等価演算子を対象に，変換はせずに，メッセージを出力することで記述の間違いを指摘する．

3.4 標準ライブラリ

標準ライブラリは，言語により定義されているものがそれぞれ異なり，一貫性がないので，各言語間で整合性をとるのが難しい．本研究では，Python の range 関数をイディオムとして扱う．

4 制御文の自動修正方法の提案

4.1 概要

コードの修正方法を考えるにあたって，まず，各言語で正しい記述を変換する方法を考え，各制御文と式の関係を明らかにする．それを基に，誤った記述を変換する方法を考える．言語の拡張性や組み合わせの数を考え，共通モデルを介した変換を提案する．共通モデルとは，対象の制御文の共通要素を一つのモデルとして表したものである．

4.2 共通モデル

図 1 に共通モデルを表す．本研究で対象とする制御文と，式の関連を表すクラス図を示している．共通モデルのインスタンスとして，IF モデル，ELSEIF モデル，ELSE モデル，WHILE モデル，FOR モデル，FOREACH モデルをそれぞれ定めた．IF，ELSEIF，WHILE モデルは，条件式 cond の関連を持ち，FOR モデルはカウンタ変数 var，初期値 initial，終了値 end の関連を持ち，終了値まで含めるかを表す，end_include を属性に持つ．FOREACH モデルは変数 var，配列 array を関連に持つ．

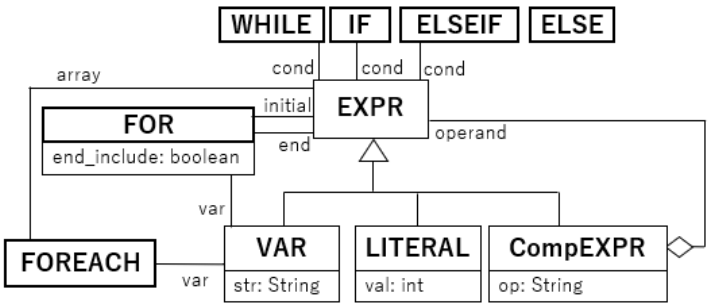

図 1 共通モデルのクラス図

4.3 共通モデルを介した変換

4.3.1 正しい記述を共通モデルへ

共通モデルに変換するには，元のコードの言語を判定したうえで，各制御文から共通の要素を抽出する必要がある．正しい記述を共通モデルに変換する方法として， 括弧の対応や，条件式の範囲を知るためには構文解析まで行う必要があるので，構文規則を元に共通モデルの要素を抽出する．コード片の字句解析，構文解析を行い，構文木の部分木から共通モデルの要素を抽出し，変換を行う．

4.3.2　誤った記述を共通モデルへ

各制御文の共通モデルが持つ，共通となる要素はすべて式である．式は使用される演算子の違いを除けば，どの言語でも記述方法は同じであるが，式との関連を表す記述方法は各言語で異なる．例えば，Ruby では for の予約語の後に，in と.. を使って，3 つの式との関連を表す．これ以降，in，;，.. などの制御文と式との関連を記述するために使われる字句を式区切り字句と呼ぶ．また，複合文の開始字句も言語によって異なる．C 言語，JavaScript，PHP では {，Ruby の if 文では then，while 文や for 文では do，Python では：である．プログラマは，式は演算子以外に誤りがなく記述できるが，制御文の予約語，式区切り字句，複合文開始字句を間違えると考えた．想定する誤りの定義を図 2 に示す．

if 文	elseif 節
・条件式を括る括弧の有無 ・複合文開始記号の誤り	・else if，elif 等の誤り ・条件式を括る括弧の有無 ・複合文開始記号の誤り

while 文	for 文，foreach 文
・条件式を括る括弧の有無 ・複合文開始記号の誤り	・for，foreach の誤り ・条件式を括る括弧の有無 ・複合文開始記号の誤り ・式の区切り字句の誤り

else 節
・複合文開始記号の誤り

図 2　本研究で扱う誤り

これらに対し，1 章で挙げた技術的課題について考える．課題 1 について，コードをいくつかの部分に分け，各部分がどの言語で記述されたかを調べ，点数化することで，言語を特定する．点数は，その言語の確からしさを表す．混同しやすい elseif 節の予約語の判定と，for 文，foreach 文の条件式は点数を高く設定する．すべての部分の判定を終えたとき，点数が最大の言語を選択し，その言語で書かれていたこととする．

課題 2 について，図 2 で誤りを定義したが，このような構文規則は存在しないので，構文解析が正しくできず，式が抽出できない．この問題は，式レベルでの構文解析を行うことで解決する．式レベルの解析とは，式区切り字句を元に，式の範囲を明らかにすることである．まず，共通モデルの要素として抽出したいものは前述の通り，すべて式であり，式の範囲がわかれば，抽出が可能となる．しかし，式区切り字句が間違っている記述は，式レベルでも正しく解析されない．そこで，式区切り字句はあらかじめ特殊な字句として定義する．式，式区切り字句，式の順で記述された場合は，式レベルの構文解析で，式，式区切り字句，式と正しく解析されるようにする．誤った記述の処理方法として，部分的に解析し，必要な情報を随時抽出する方法を考えた．文全体を見て，どういう構造か調べるのではなく，間違えやすい箇所を部分的に解析する．部分的な解析とは，その箇所がどの言語で書かれているかを照合することである．正しく照合ができたとき，共通モデルに順次，変換していく．同時にその部分が，どの言語で書かれているかを判定する．共通モデルに変換できたとき，そのコードは本研究の対象の文であり，言語の判定も完了する．照合する項目を解析項目，その解析項目で判定される言語の点数を言語情報とする．各部分の解析は，図 2 を基に項目を作成する．例として，for 文の条件式の解析項目と，それぞれの解析項目における言語情報を表 3 に示す．また，誤った for 文の解析例を図 3 に示す．

表 3　for 文の条件式の解析項目

解析項目	言語情報
var=*expr*; *var*<*expr*; *var*++	C,PHP,JS +3
var=*expr*; *var*<*expr*; *var*=*var*+**1**	C,PHP,JS +3
var=*expr*; *var*<=*expr*; *var*++	C,PHP,JS +3
var=*expr*; *var*<=*expr*; *var*=*var*+**1**	C,PHP,JS +3
var (**in**|**of**|**as**) **range**(*expr*)	Python +3
var (**in**|**of**|**as**) **range**(*expr*,*expr*)	Python +3
var (**in**|**of**|**as**) *expr*..*expr*	Ruby +3
var (**in**|**of**|**as**) *expr*...*expr*	Ruby +3

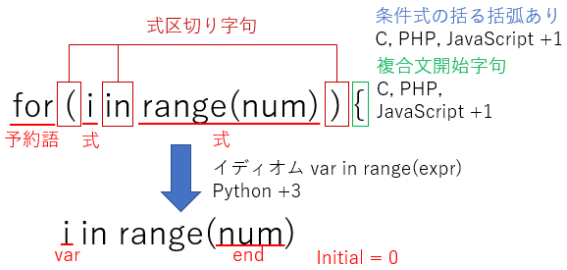

図 3　誤った for 文の解析

4.3.3　共通モデルから目的言語のコード

目的言語のコードへの変換は，あらかじめ，目的言語で書かれたテンプレートを用意し，そのテンプレートに共通モデルの要素をあてはめる．Ruby のテンプレートを表 4 に示す．FOR モデル 1 は終了値を含む FOR モデルからの書き換えのテンプレート，FOR モデル 2 は終了値を含まない FOR モデルからのテンプレートである．

表 4　Ruby のテンプレート

	Ruby のテンプレート
IF モデル	**if** *cond* **then**
ELSEIF モデル	**elsif** *cond* **then**
ELSE モデル	**else**
FOR モデル 1	**for** *var* **in** *initial*...*end* **do**
FOR モデル 2	**for** *var* **in** *initial*..*end* **do**
FOREACH モデル	**for** *var* **in** *array* **do**

5　自動修正ツールの実現

5.1　設計

本ツールは，TEBA[5] を用いて，共通モデルを介した変換や言語判定を実現する．TEBA の役割は， 文法的に正しいと限らないコード片に対して字句解析，構文解析を行うことと，各言語のコードと共通モデル間の変換を行うことの 2 つである．1 つ目の役割について，TEBA は字句解析で得られる字句系列に，精度の低い構文解析をしたあと，精度を向上させることができる部分を徐々に構文解析していく方法を用いる．これにより，正しいと限らないコード片に対して，式レベルの解析ができる．2 つ目の役割について，TEBA の書き換えルールを用いて実現する．書き換えルールとは，書換え前の字句列の状態と書換え後の字句列の状態の記述されたファイルである．これを書き換え用のプログラムで読み込むことで，コード片の属性付き字句系列から共通モデルの属性付き字句系列の書き換えを実現する．解析項目を書き換え前の字句列に定義する．正しく照合された場合の，書き換え後の字句列は，必要に応じて，言語情報や共通モデルの要素を定義する．

5.2　実現

変換対象のコード 1 行と目的言語を JSON 形式で入力すると，コードの記述言語と目的言語に変換したコードを JSON 形式で出力する自動修正ツールを，Perl を用いて実現した．提案したツールを用いて，Visual Studio Code で編集中のコード 1 行を修正するプラグインを実現した．Visual Studio Code で，Listing1 の記述を C 言語に変換した例を図 4 に示す．また，提案したツールを WebAPI としても実現した．

図 4　Listing1 の変換

6　考察

本研究では，変数の型は解析していない．配列の要素を順次処理する反復において，共通モデル FOREACH の要素 ARRAY となる変数に，整数値など，反復可能でない値が代入されていても変換する．その場合は本来コンパイルエラーになるので，修正しなければならないが，そのまま変換を行う．型情報を踏まえた変換をしていないことが改善点である．

言語の拡張について，元の記述から共通モデルに変換する書き換えルールを追加し，目的言語のテンプレートを作成することで拡張が可能である．制御文以外の構文の拡張について，本研究ではコード 1 行単位で変換をするので，コード 1 行からコード 1 行への変換で整合性が取れないものに関しては拡張は難しい．幅広く対象の拡張をするためには，前の文脈も考慮する必要がある．誤りパターンの拡張について，本研究で定義した誤りパターン以外にも，式の誤りや型の誤りなど，様々な誤りが考えられる．本研究の発展として，本研究で定めた誤りパターン以外の誤りについて分析し，修正方法を実現していくことがあげられる．

7　おわりに

本研究では，制御文を対象にして，既習言語の知識に起因する誤りの自動修正方法を提案した．既習のプログラミング言語と学習中のプログラミング言語の文法が混同することにより起こりうる，他言語での記述や複数言語の文法が混在した記述に対して自動で修正する．ソースコード編集中に改行をした際に，制御文か判定し制御文ならばコードを解析し，何の言語で書かれていたか指摘し，目的言語の正しい記述に修正する．

今後の課題として，言語判定の妥当性，変換の整合性，対象の拡張性などを踏まえた上での手法・ツールの改良，実際にプログラミング演習などで使用し，実用性を検証することが挙げられる．

参考文献

[1] 間辺 広樹，長島 和平，並木 美太郎，長 慎也，兼宗 進：C の学習経験を持つ高校生への Python の授業導入事例，情報教育シンポジウム論文集，2019，pp.256-262(2019).

[2] 坂本 一憲，大橋 昭，太田 大地，鷲崎 弘宣，深澤 良彰：UNICOEN：複数プログラミング言語対応のソースコード処理フレームワーク，情報処理学会論文誌，Vol.54，No.2，pp.945-960(2013).

[3] Lachaux, M., Roziere, B., Chanussot, L. and Lample, G.:Unsupervised Translation of Programming Languages, arXiv:2006.03511v3(2020).

[4] Alon Zakai, Tim Dawborn, Max Shawabkeh, et al: Main - Emscpipten 1.39.20 documentation, available from <https://emscripten.org/> (accessed 2020-12-28).

[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).