5L-4

ソースコード内の文字出現率を利用したコードクローン判別法

垣谷 広輝† 安藤 優作† 増田 智樹† 平山 雅之† 菊地 奈穂美‡

日本大学† 沖電気工業株式会社‡

1. はじめに

ソフトウェアの保守性を阻害する要因としてソースコード内のコードクローン（以降，クローンと表記）の存在が問題視されている．クローンは，コード内の処理をコピー&ペーストすることで生まれる類似コード断片(コードブロック)である．保守性を向上させるためにはクローンを早期に検出し，内容確認や場合よっては除去するなどの対策を行う必要がある．

従来，クローンは主に構文解析を利用する方法によって検出されてきた．この方法では文法的に誤りのあるコードへの適用が難しいといった問題があった．このため我々は，クローンの検出において構文解析を用いず，コードブロック内の処理の形状面から類似性に着目したクローン検出法(CICD:Code Image Clone Detection method)の開発を進めている．

本論では，CICD の中で利用している文字出現率を用いたコードブロックの類似性判定法について，簡易な実験結果も交えて報告する．

2. CICD の概要

本章では CICD 法のおおよその流れを紹介する．CICD 法では次の Step1 から Step4 の 4 ステップによってクローンの検出を行う[1].

Step1:対象ソースコードを処理ブロックに分割.

Step2:各ブロック内の総文字数，総行数を利用したクローン候補ブロックの特定.

Step3:候補ブロックの各行ごとの文字数比較によるクローンブロックを特定.

Step4:特定したクローンブロックをクローンとして確定するためにブロック内の顕著な特徴を抽出して確定診断を行う.

図 1 は Step3 までの処理で特定されるクローンブロックの様子を示す．図 1 の例では，コード A，コード B の 2 つのコードについて，それぞれ 3 つの処理ブロックに分けられる．そして，そのうちの①と⑥，②と④，③と⑤が形状的に

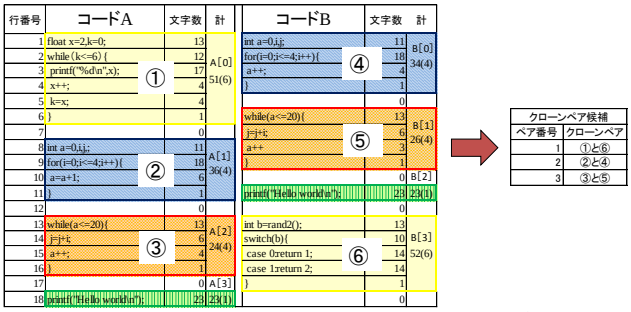


図 1 CICD 法によるクローンブロック候補抽出

類似するクローン候補ブロックとして特定できることを示している．

本論ではこのようにして抽出されたクローンブロック候補に対し，各ブロック内で出現する特定文字の出現率を利用して，より高い精度でクローン判定を行うことを目的としている．

3. 特定文字出現頻度を利用したクローン判定

Step4 では，クローン候補となるコードブロック内の特定文字の出現頻度を比較し，クローンであるかどうかを高精度で判定する．特定文字には，コードブロック内の処理特徴をあらわす文字を利用する．特定文字は対象言語の特徴から抽出されるものもある．

具体的には，各ブロック内での論理構造や演算処理面の処理特徴を表す文字として，C 言語の場合であれば，次のものを候補として考える．

特殊文字[記号]：<, >, +, −, /, %, &=, (,) など

アルファベット ：for, while, if, else, do, switch, case など

例えば，図 1 で候補として抽出したブロック③と⑤，およびブロック②と④の場合(図 2)，それぞれの特殊文字の出現数は表 1 のようになる．この表をみると，どちらのブロックの組も出現する特殊文字の数に大きな差は見られず，処理内容や論理構造が同じであれば，特定文字の出現数もほぼ同じになると考えることができる．

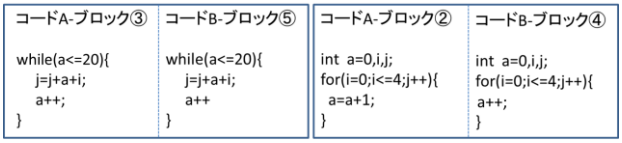


図 2 特定文字を利用したブロック類似評価

Code clone candidate detection technique with characters' appearance ratio

†Nihon University

‡Oki Electric Industry Co., Ltd.

表1　特定文字の出現数評価結果

		<	+	=	;	f	w
ブロック	A③	1	4	2	2	0	1
	B⑤	1	4	2	2	0	1
ブロック	A②	1	3	4	4	1	0
	B④	1	4	3	4	1	0

4. 特定文字の選択実験

前章に示したように特定文字を利用して，クローンブロックの処理特徴を評価する場合，特定文字としてどの文字を採用するかが重要となる．このため，ここでは，実際のC言語のオープンソースを評価材料として，

a. どのような文字が個々のブロック内に含まれるか，
b. その文字の中でどの文字を評価する特定文字にすべきか

を決定するために実験を行った．

4.1　実験方法

評価材料は下記のオープンソースの2ファイルから一部を無作為に抜粋したものを使用する．

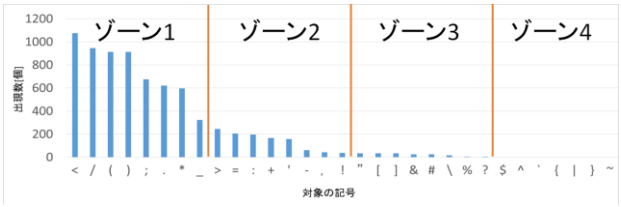
・Borland C++ Compiler 5.5　stdlib
・indent ver2.2.9　intl

実験ではこれらのファイルを CICD の Step1 を適用し処理ブロックに分割し，各ブロック内にどのような文字が含まれるかをカウントした．この際，ブロックは if,while などの論理構造をもつ複雑な処理ブロックと，それらを持たない単純構造の処理ブロックの 2 タイプに分けて評価を行った．

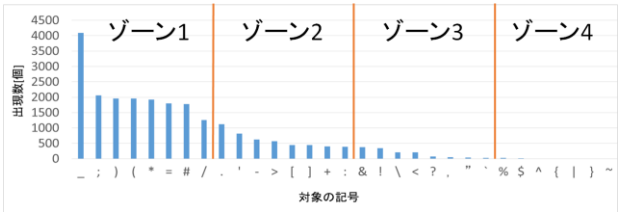
4.2　評価結果

(a) 特殊文字[記号]

図 3(1)は stdlib の複雑ブロック内の記号出現数を，図 3(2)は intl 内のブロック全体における記号の出現数を示したものである．この 2 つの図はそれぞれのブロック内での記号の出現数を頻度順に表しており，出現頻度別に上位から四分の一ずつにゾーンで区切ってある．この中で第 1 ゾーンはどのようなブロックでも比較的多く出現する記号であるため処理特徴を表現する文字としては適していないと考える．また，第 3，4 ゾーンは出現が非常に希なものであり，個々の処理ブロックに含まれない場合もあるため処理特徴を評価する記号としては適していないと考えられる．これらを考えるとブロックの特徴を表現する文字としては第 2 ゾーンに含まれる文字を採用することが良いと考えられる．このゾーンには,処理内容の加算+や減算-比較>などを表現する文字が含まれており，これらをブロックの処理特徴を評価する文字として利用することが有効と考えられる．



(1) stdlib 複雑ブロック

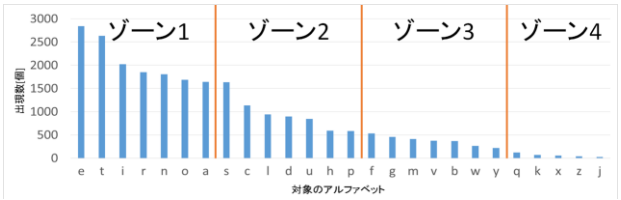


(2) intl 全体

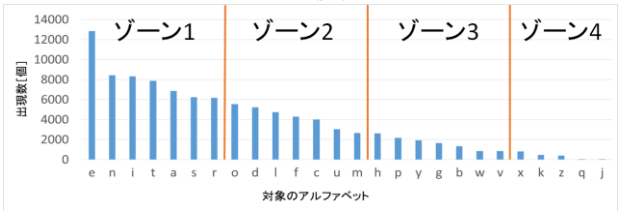
図 3　記号の出現頻度

(b) アルファベット

図 4(1)(2)にアルファベットの出現頻度を示す．記号と同様に 4 つのゾーンに分けると，この場合も第 2 ゾーンに含まれる文字を評価用の特殊文字に採用するのが適切と考えられる．このゾーンには，for,while,case などに含まれる f,h,c などが入っており，これらの文字は処理の特徴をとらえる文字として利用することが有効と考えられる．



(1) stdlib 複雑ブロック



(2) intl 全体

図 4　アルファベット(小文字)の出現頻度

5. まとめ

本論ではクローンブロックに含まれる特定文字によるクローンの判定精度向上法のアイデアを紹介した．また，簡易実験により判定時に利用すべき特定文字候補の絞り込みについて検討した．今後，この結果をさらに精査し，特定文字による判別の精度を上げていきたい．

参考文献

[1]垣谷 広輝，安藤 優作，平山 雅之，菊地 奈穂美，"ソースコードを構成する処理ブロックの特徴に着目したコードクローン推定技術"，情報処理学会研究報告ソフトウェア工学研究会，Vol.2013-SE-182(29)，2013.